國立交通大學

資訊科學與工程研究所

碩士論文

以人為本的存在感知

即時通訊與通訊錄同步系統的設計與實作

Design and Implementation of a User-Centric Presence-aware Instant Communication and Address Book Synchronization System

研究生：施威年

指導教授：曾建超　教授

中華民國　九十五　年六月

以人為本的存在感知即時通訊與通訊錄同步系統的設計與實作

Design and Implementation of a User-Centric Presence-aware Instant Communication and Address Book Synchronization System

研 究 生：施威年　　Student: William Shy

指導教授：曾建超　　Advisor: Chien-Chao Tseng

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2006

Hsinchu, Taiwan, Republic of China

中華民國九十五年六月

國立交通大學

研究所碩士班

論文口試委員會審定書

本校　資訊科學與工程　研究所　施　威　年　君

所提論文: 以人為本的存在感知及時通訊與通訊錄同步系統的設計與實作

Design and Implementation of a User-Centric Presence-aware Instant Communication and Address Book Synchronization System

合於碩士資格水準、業經本委員會評審認可。

口試委員：

指導教授：曾建超

所　長：

中 華 民 國 九十五 年　六　月　十六　日

以人為本的存在感知
及時通訊與通訊錄同步系統的設計與實作

研究生：施威年　　指導教授：曾建超

國立交通大學資訊學院資訊科學與工程研究所

摘　要

網路科技的演進使得使用者可以透過各式各樣的通訊系統與他人聯繫，如電子郵件，或是像 MSN Messenger、Yahoo Messenger、Skype 之類的即時訊息系統。由於同時存在著這麼多不同的通訊方式，使用者在網路上的聯絡人很可能使用著不同的通訊系統。再者，每一種通訊系統都用它自己的方式來維護自己的聯絡人清單。於是對於使用者來說，在各種通訊系統的聯絡人清單中找出某個特定的聯絡人、得知該聯絡人的上線狀態、以及與該聯絡人透過適當的方式聯繫就成為一種困擾。

在這篇論文中，我們設計並實作出一套「以人為本的存在感知及時通訊與通訊錄同步系統」，以幫助使用者管理跨越各種通訊系統的聯絡人清單、獲知聯絡人的上線狀態，以及方便地透過最適當的通訊系統與聯絡人聯絡。並且，此系統也具備與使用者在不同裝置上的系統通訊錄同步之能力。

最後，有別於其他的即時通訊互通平台，我們的系統是以使用者為中心，並加上即時通訊互通機制。因為它提供了一個方便友善的介面，讓使用者辨別每個聯絡人的識別身份與上線狀態—因為每個聯絡人在不同的即時通訊系統中可能有不同的識別身份與上線狀態。使用我們的系統，使用者可以方便地察看聯絡人的上線狀態、以群組方式管理聯絡人，並方便地與某位聯絡人用適當的方式通訊。除此之外，我們的系統也提供一個公開的介面，讓第三方系統可以與我們的系統交互合作，來發展出其他需要即時通訊能力之新系統。

Design and Implementation of a User-Centric Presence-aware Instance Communication and Address Book Synchronization System

Student: Willian Shy Advisor: Dr. Chien-Chao Tseng

Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

ABSTRACT

The advance and pervasion of Internet technologies make it possible for Internet users to communicate with each other via various communication systems, such as e-mails or Instance Message Systems, for examples MSN Messenger, Yahoo Messenger, Skype and many others. With so many communication methods existing, it is possible that the contacts of an Internet user may use different communication systems. Furthermore, each of the communication systems maintains the contact list of a user in its own way. Therefore, it is very troublesome for a user to find a contact from the contact lists maintained by various communication systems, determine the presence status of the contact, and communicate with the contact via an appropriate communication system.

In this thesis, we design and implement a “User-Centric Presence-aware Instance Communication and Address Book Synchronization System” that can help a user to manage his contact lists across various communication systems, determine the presence status of a contact, and click to communicate with the contact through the most appropriate communicate system. Furthermore the platform also enables the user to synchronize his address books across various devices.

Finally, unlike other inter-communication platforms of different IMSs, our platform is user-centric, in addition to the inter-IMS communication mechanism, because it also provides a friendly interface for a user to gather the identities and presence statuses of each contact, which may have different identities on different IMSs. Therefore, using our system, a user can easily check the presence of contacts, manage contacts by grouping, and communicate with a contact by an appropriate method conveniently. Beside, our system also provides a public interface for other third-party systems to inter-work with our platform for novel Internet services that require instant communication services.

致　謝

本論文承蒙指導老師 曾建超教授的多方指導及殷殷教誨，得以順利完成，在此致上無限的謝意。

此外，感謝所有我的師長、同學們。感謝你們的熱心幫忙與耐心教導。感謝我的朋友們，在我面對壓力時願意傾聽並給予鼓勵。

最後，感謝我的家人，特別是養育我、栽培我的父母，讓我全心全意地完成學業與論文，在此獻上我最衷心的感謝。

目　錄

圖 目 錄

表 目 錄

第一章 緒論

1.1 研究動機

近年資訊科技發展迅速，進步一日千里。自電晶體的發明以來，短短數十年，計算設備已經從充塞整個房間的大型電腦，變成可以握在手心，放進口袋的隨身裝置。而生產技術的進步也帶來成本的快速下降，更讓資訊科技完全融入生活，而網際網路的發明，更對我們的生活帶來全面的影響。以即時通訊(Instant Messaging, IM)來說，從 UNIX 時代之 IRC(Internet Relay Chat)，1996 年的 ICQ，到近年來崛起的 MSN Messenger、Yahoo! Messenger、Skype，各種通訊軟體百家爭鳴，即時通訊已經全面進入了我們的生活。而近年來的發展，即時通訊也開始跨出電腦網路，從早期的純文字交談，到語音、視訊，最近甚至可以直接接通電話網路—我們可以坐在電腦前，透過電腦麥克風，與使用手機的朋友交談。

但是技術的多元發展也帶來了問題，每個人都有可能使用著不同的系統。某些聯絡人使用 MSN、而另一些聯絡人則使用 Skype，當聯絡人的數量持續成長，要找出與某個聯絡人溝通的方法就變得繁瑣複雜。以一個情境來描述：

> 假設今天在家裡我有急事要與某甲聯絡，我打開 MSN Messenger 看看某甲有沒有在線上，又想起他用的是 Skype，打開 Skype 又發現他不在線上，於是現在又要打開通訊錄找他的 e-mail，打開了通訊錄，發現某甲的聯絡資訊放在公司的電腦上…

我們是不是能有一個更簡便的方法可以輕鬆找到聯絡人呢？

而從另一方面來說，即時通訊的普及也開啟了更多聯繫人與人的管道，除了傳統的電話、住址，更新一點的 e-mail、即時通訊帳號也漸漸成為通訊錄的內容之一。如果能發展一個方法，讓通訊錄與即時通訊系統整合，我們可以在通訊錄裡一目了然地看到好友的上線狀態，也可以方便地與這位好友聯繫。而這套系統也可以與傳統的通訊錄整合，同步通訊錄的內容。不只本地端通訊錄與我們的系統同步，甚至也可以同步多部電腦上的通訊錄內容。

這樣一個系統的設想與製作，就是本論文的動機。

1.2 研究目標

我們的目標是發展一個整合即時通訊上線狀態之聯絡人管理系統。目標與主軸有二：(1)使用者上線狀態感知之即時通訊，以及(2)聯絡人管理與同步。

(1) 使用者上線狀態感知之即時通訊

在第一個目標上，我們希望可整合多種不同的即時通訊系統。在我們的系統裡可以一目了然地看到聯絡人的各種上線狀態，並可以方便地直接使用對應的通訊方式與聯絡人聯繫。舉例而言，今天看到某甲的 MSN Messenger 上線，我們可以直接命令系統「要與某甲以 MSN Messenger 聯繫」，而系統就會幫我們打開某甲的交談視窗。今天某乙所有即時通訊軟體都不在線上，我們也可以選擇要與某乙手機聯繫，而系統就會幫我們打開 Skype，並使用 SkypeOut 撥號到對方的手機。在本論文裡選擇 MSN Messenger 與 Skype 作為整合的目標，並且在設計上保有擴充性，未來可以整合更多其他的系統。

我們的系統是以聯絡人為中心，整合該聯絡人使用的即時通訊系統至同一聯絡人項目。比如說某聯絡人使用MSN Messenger以及Skype，則在系統中就可以同時獲知該位聯絡人的MSN Messenger以及Skype上線狀態；同時，我們的系統只要求使用者安裝有對應之即時通訊系統之客戶端程式，而不需要使用者的所有聯絡人進行任何特定動作（如安裝某個程式、加入某個特定聯絡人）。而不同於2.2中描述之其他系統。

	Name	Phone #	MSN	Skype
	Ronaldo	031234567	Online	Offline
	Beckham	037654321	Offline	Offline
	Kahn	045213568	Away	Away
	Zidane	063593470	Online	Busy

圖1 聯絡人通訊錄與上線狀態之整合

(2)聯絡人管理與同步

在第二個目標上，我們希望能提供群組式的聯絡人管理機制，並與本地端的通訊錄同步。假設我們在系統裡新增了數個聯絡人，並把這些聯絡人建立一個新群組，這些改變都會反映到本地端的通訊錄軟體上。同樣地，而在本地端所做的任何動作，也會反映到我們的系統裡。如果是多部電腦的情況，便可以同步多部電腦上的通訊錄內容。我們實作的目標是 Windows Address Book(在中文版 Microsoft Windows XP 上稱為「通訊錄」)，當然在設計上也是希望保持擴充的可能性，以期未來能支援不同的通訊錄系統。

1.3 章節概要

在第一章裡，描述了本論文之動機與目標；第二章則包含了背景與相關系統之研究與比較。第三章根據我們的目標來訂定詳細的系統需求；第四章則開始設計系統的架構，並說明系統各個部分元件的實作過程以及使用的相關技術，並舉例說明元件的互動方式。第六章則實際展示了系統開發的成果。最後一章，闡述本論文之結論以及未來期望之發展方向。

第二章 背景與相關研究

2.1 背景

即時通訊是一種奠基於網際網路之訊息傳遞服務。不同於電子郵件，即時通訊的訊息傳遞是即時的。使用者只要有該系統的客戶端軟體，就可以與同樣使用客戶端軟體之使用者聯繫。使用者在本地客戶端輸入的訊息，會透過網際網路傳遞，即時出現在對方的客戶端軟體上，使用者即可透過這樣的訊息交換機制來進行溝通。大部分之即時通訊系統提供了聯絡人上線狀態(presence awareness)之功能—軟體提供聯絡人清單，使用者可以在清單上看到聯絡人之上線狀態，與是否可與之交談。

即時通訊之發展濫觴於 70 年代，UNIX 系統間流行的 IRC，而真正開始流行則是始於 1996 年以色列軟體公司 Mirabilis 推出之 ICQ。自此以後即時通訊發展快速，近年已經與 e-mail、WWW 同成為網際網路服務之主流。許多大型軟體公司紛紛投入，包括 MSN Messenger、Yahoo! Messenger 等即時通訊軟體，因而呈現百家爭鳴之狀態。早期的即時通訊都只提供文字訊息交換服務，而近年來由於網路頻寬以及多媒體技術之進步，更進一步發展出語音與視訊之對談服務，甚至也跨出了網際網路，出現了可以直接與傳統電話網路通訊的服務：如 Skype。

而在聯絡人管理方面，在 e-mail 的普及之下，使用者有了在電腦上管理聯絡人的需求，通訊錄也變成各作業系統內建的基本功能。同時在網頁電子郵件(Web mail)服務發展上，各家服務供應商多半也提供了線上的聯絡人管理功能。如果把即時通訊之聯絡人清單也視為一種通訊錄，則可以說是通訊錄無所不在，但是它們卻內容各異，也缺乏整合。

在百家爭鳴的情況下，使用者面對的是數種不同的通訊錄，每一個可能都有上百筆互相重疊的聯絡人資料。即時通訊與聯絡人管理的本意是便利，在這些通訊錄裡要尋找出特定的聯絡人，反而成為一種負擔。

2.2 即時通訊系統之存在感知

在即時通訊系統之存在感知方面，我們選擇文字交談即時通訊服務中，具代表性的MSN Messenger；以及語音交談即時通訊服務中，具代表性的 Skype 做介紹。同時這兩種即時通訊服務也是本論文選擇實作的目標。

2.2.1 Microsoft MSN Messenger

微軟(Microsoft)公司推出之MSN Messenger服務是現今即時通訊軟體主流之一。除了MSN Messenger客戶端軟體之聯絡人清單之外，MSN Messenger之聯絡人存在感知也整合至同為MSN旗下之Hotmail服務。使用者在登入Hotmail以後，可以看到在自己的聯絡人上線狀態(請參考圖 2)。

這個服務是完全基於用戶端的。Hotmail 在網頁裡使用 VBScript，透過 ActiveX 取得 MSN Messenger 客戶端之聯絡人狀態，並顯示於客戶端的網頁上。這樣的作法，缺點在於無法跨瀏覽器，因為其使用的 VBScript/ActiveX 僅能在 Internet Explorer 上執行。除此之外 MSN Messenger 也沒有提供一個開放的方法，讓第三方獲取 MSN Messenger 使用者的上線狀態。換言之，這是一個完全封閉的服務。

圖2 Hotmail之聯絡人上線狀態顯示

2.2.2 Skype

Skype亦是現今之主流即時通訊軟體之一，相對於其他以文字服務為主軸之即時通訊系統，Skype強調的是PC-to-PC與PC-to-Phone的語音對談服務。不同於前述之MSN Messenger，Skype用另一種較為開放的方式提供了使用者之上線狀態提示。Skype提供的機制稱為Skype4Web，Skype4Web提供一組API (Application Program Interface)，讓第三方可以存取系統內使用者之上線狀態。使用者在Skype的偏好設定中開啟此項功能後，第三方就可以透過這組API去獲取使用者之上線狀態，

圖 3 顯示了此功能之運作機制。

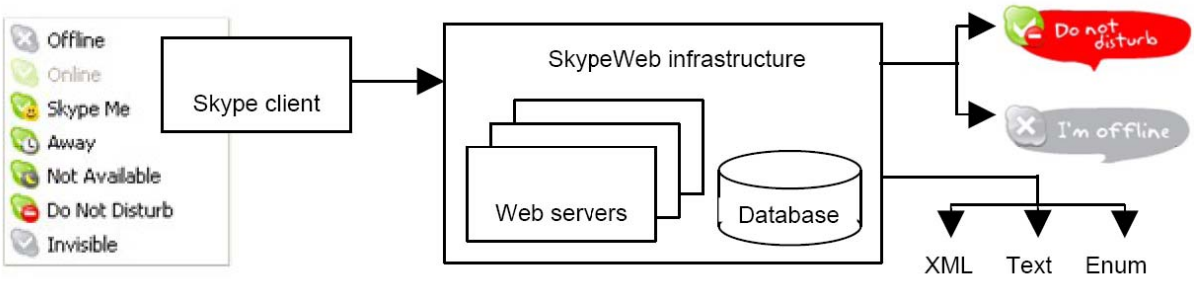


圖3 Skype4Web 架構圖

Skype自己也利用此機制提供了稱為Skype buttons的服務，使用者可以用此功能產生一組HTML程式碼，程式碼裡包含一張圖片，此圖片會根據使用者的上線狀態而變化。使用者可以將這組HTML碼加到電子郵件簽名裡，或是個人網頁上，其他人就可以看到使用者的上線狀態，並可直接點選圖片開取與使用者的對話視窗，實例請參考圖 4。

圖4 Skype buttons

2.2.3 libGAIM

libGAIM 是一個支援多種通訊協定之即時通訊函式庫，並可將多個即時通訊系統之聯絡人整合於一列表。但 libGAIM 的聯絡人整合是將所有即時通訊系統的聯絡人並列在同一份列表中，當使用者點選欲通訊的聯絡人時，再使用該聯絡人使用的即時通訊系統與之通訊。如果某一聯絡人同時使用 MSN Messenger 以及 ICQ，則在 libGAIM 中會將其視為兩個不同的聯絡人。目前 libGAIM 只支援文字交談，且不支援 Skype 通訊協定。

	Name	Status	Type
	Zidane	Online	MSN Messenger
	Ronaldo	Offline	MSN Messenger
	Zidane	Away	AOL Messenger
	Kahn	Online	AOL Messenger
	Totti	Offline	Google Talk

圖5 libGAIM的聯絡人上線狀態整合概念

2.3 聯絡人管理相關系統

2.3.1 離線聯絡人管理系統

因為網路的普及，現在所有主流作業系統幾乎都內建了聯絡人管理的功能。這些聯絡人管理程式其資料只儲存於本地端，使用者只能在自己的電腦上存取，故稱為離線聯絡人管理系統。這些程式通常提供使用者管理聯絡人的姓名、地址、電子郵件帳號等，也通常提供第三方的呼叫，讓其他程式可以使用作業系統的聯絡人資料。圖 6為Microsoft Windows內建之Windows Address Book，Microsoft Outlook以及Outlook Express都可使用Windows Address Book作為電子郵件通訊錄。

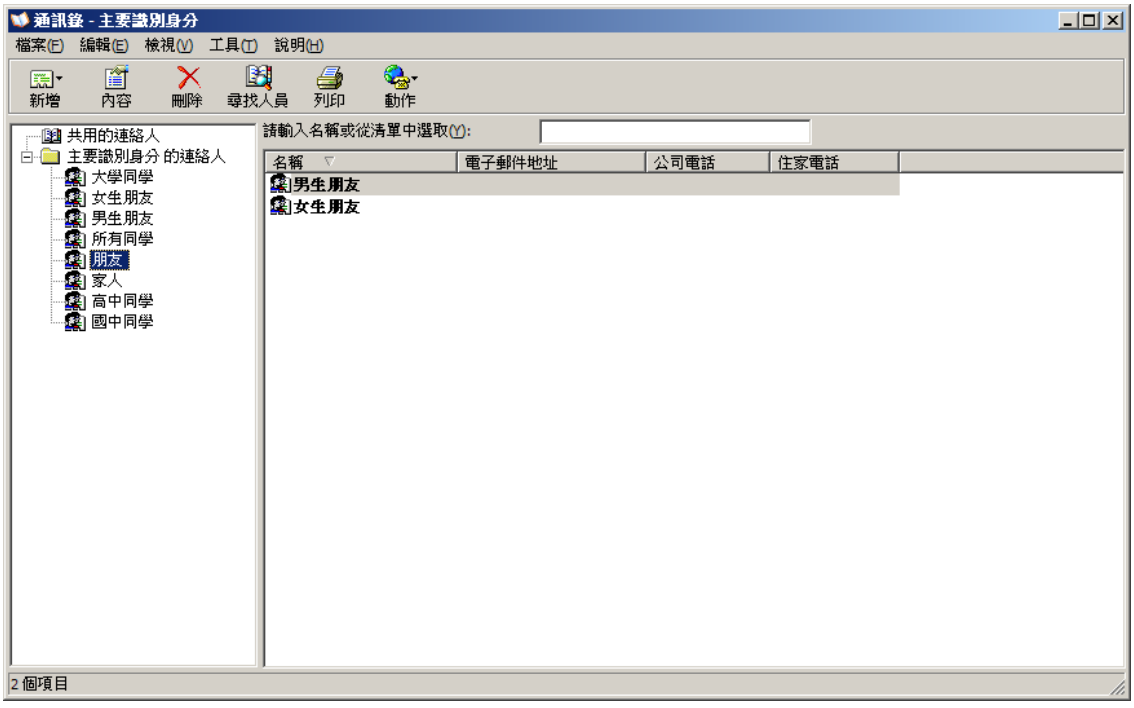


圖6 Windows Address Book

這些離線的通訊錄程式通常可以讓使用者將聯絡人資料匯出為各種格式的檔案，而使用者可以將這些檔案匯入其他的軟體，以達成聯絡人資料之轉移與備份功能。然而離線的通訊錄系統，資料只能在本地端存取。在現今人人擁有多台電腦的環境下，各電腦裡的通訊錄內容並不同步，使用者必須花費多餘的心力去管理各台電腦上的聯絡人資料庫。

2.3.2 線上聯絡人管理系統

許多入口網站都提供 Web mail 服務，而這些服務大都提供線上的通訊錄功能。這些服務其聯絡人資料庫儲存於網際網路上，故稱為線上聯絡人管理系統。

這些線上的通訊錄採用網頁介面，由於資料儲存在網路端，可以達成讓使用者在每台電腦都存取到同樣聯絡人資訊的需求。但是也因為是附加的服務，所以只能與自家的 Web mail 系統共用，也無法與本地端的通訊錄互通。雖然有些通訊錄提供與即時通訊整合的功能，如前述的 Hotmail，但也只支援自家的即時通訊系統，而無法整合各個不同的即時通訊上線狀態。

第三章 系統功能需求分析

根據我們的背景動機，以及參考過相關的系統以後。我們理想中的系統，應該滿足下列的功能需求。

3.1 伺服器端系統之功能需求

1. 基於網頁的存取方式

我們希望能以 Web 介面呈現系統。假設使用者現在並不在自己的電腦上存取聯絡人資訊，也許使用者使用的電腦並沒有安裝我們的客戶端程式。以網頁存取，即使使用者沒有安裝我們的客戶端程式，至少也可以透過瀏覽器來存取自己的聯絡人資料庫。

2. 存取權限管理

聯絡人資料具有一定的私密性，在伺服器端必須提供存取權限管裡的機制。讓使用者只能存取自己的聯絡人資料，而不能看到其他聯絡人的資料。所以伺服器端必須提供使用者之身份認證機制：使用者必須以帳號密碼登入系統後，才可存取到聯絡人資料，無論是網頁介面或是客戶端程式。

3. 聯絡人管理

我們希望提供群組式的聯絡人管理功能，包括新增、刪除、修改聯絡人資料。以及把多個聯絡人劃分為群組。系統必須把這些資訊儲存在伺服器端，並在使用者透過或客戶端登入系統後可以加以存取。使用者可以透過網頁介面管理自己的聯絡人，或客戶端程式根據這些資訊來同步客戶端的聯絡人資料庫。

4. 聯絡人上線狀態資訊

聯絡人的上線狀態也是本系統的主要功能之一。伺服器端必須記錄聯絡人的上線狀態，並提供一個機制讓客戶端程式能提供最新的聯絡人上線資訊給伺服器，而使用者可以透過網頁介面獲取伺服器端所儲存的聯絡人上線狀態。

5. 即時通訊

系統提供與即時通訊的整合。當使用者在網頁介面看到聯絡人的上線資訊後，可以選定聯絡人以及通訊的方式。當使用者選定要通訊的對象以及方式，系統會呼叫客戶端對應的程式來幫使用者建立與該對象的通訊管道。

3.2 客戶端系統之功能需求

1. 客戶端聯絡人資料管理

客戶端程式必須有能力存取客戶端通訊錄軟體的聯絡人資料庫。包括聯絡人資料的獲取、新增、刪除、修改等。如果通訊錄軟體支援群組式的管理機制，客戶端也具備提供群組的存取管理功能。

2. 聯絡人上線狀態存取

客戶端程式必須有能力與客戶端的即時通訊軟體溝通。並獲取聯絡人的即時通訊上線狀態。客戶端可以透過伺服器端提供的機制，將這些聯絡人上線狀態傳遞至伺服端。

3. 聯絡人資料之同步

客戶端程式必須能夠同步客戶端與伺服器端的聯絡人資料。根據聯絡人的修改狀況，判斷出該選用本地端或伺服器端的資料，讓兩端的聯絡人資料庫都保持在最新的狀態。客戶端程式可根據本小節第 1 點所描述的功能來管理本地端之聯絡人資料，並利用伺服器端提供之機制來管理客戶端之聯絡人資料。

4. 提供友善的使用者介面

本系統的本意在讓使用者可以方便的獲知聯絡人的狀態，並與聯絡人通訊，同時能夠管理聯絡人。所以必須提供直覺的方式，讓使用者可以一目了然得知聯絡人的上線狀態，並且方便地與該聯絡人通訊。

3.3 系統功能設計與實作上之考量

本系統旨在整合，期望能支援多種的即時通訊以及通訊錄系統，所以在系統的設計上希望能盡量保持彈性以及擴充性，以期在未來發展上能方便地開發出對於其他系統的支援功能。同時藉由模組化的設計，將各功能單元盡量分開，以方便程式的維護以及改進。未來在發展更多元的系統功能時，對於系統內原有元件之影響衝擊能降至最低。

另外，我們也期望將來能應用本系統來開發出其他的系統，所以系統必須能與其他系統並用，在設計上希望能提供一方便之介面讓外部系統存取，與其他系統合作—本系統既可獨立運作，也可以成為其他系統的一個成員。

3.4 系統應用實例示意

以下就以幾個情境來說明我們構想中的系統，使用時的情況：

使用者的電腦上已經安裝並執行了我們的客戶端程式，使用者打開瀏覽器，登入系統以後，可以看到聯絡人資料的畫面。聯絡人已經根據使用者的設定，分門別類劃分成群組。

現在使用者想找一位高中同學甲。使用者根據已劃分的群組，打開「同學」群組，再找到底下的子群組「高中同學」，開啟該群組後就可以看到所有自己的高中同學。使用者在聯絡人列表裡看到甲的 MSN Messenger 為上線狀態，便要求系統要與聯絡人甲使用 MSN Messenger 通訊。系統接收到來自使用者的命令，會呼叫使用者端的 MSN Messenger，並開啟 MSN Messenger 之交談視窗，與甲建立連線，使用者就可以開始與甲交談。

使用者結束與甲的交談後，又想與乙交談。使用者根據乙的群組找到聯絡人乙，發現聯絡人乙目前沒有任何即時通訊程式上線。在這樣的情況下，使用者就要求與聯絡人乙用電話聯繫，此時系統就根據伺服器端儲存的乙的個人資料，調出電話號碼，並打開使用者的 Skype 客戶端，以 SkypeOut 撥打對方之電話號碼。

結束會談後，使用者想起今天收到一張新名片，使用者在系統裡裡建立一個新聯絡人，並把新的聯絡人丙資料輸入進去。第二天，使用者到達辦公室，辦公室電腦也裝載了我們的客戶端程式，使用者打開公司電腦的通訊錄，發現昨天新加入的聯絡人丙資料已經正確無誤地出現在辦公室的電腦裡。

第四章 系統設計與實作

4.1 系統架構概觀

本系統採client-server架構，伺服器端與客戶端透過網際網路溝通。圖 7為本系統之架構圖。

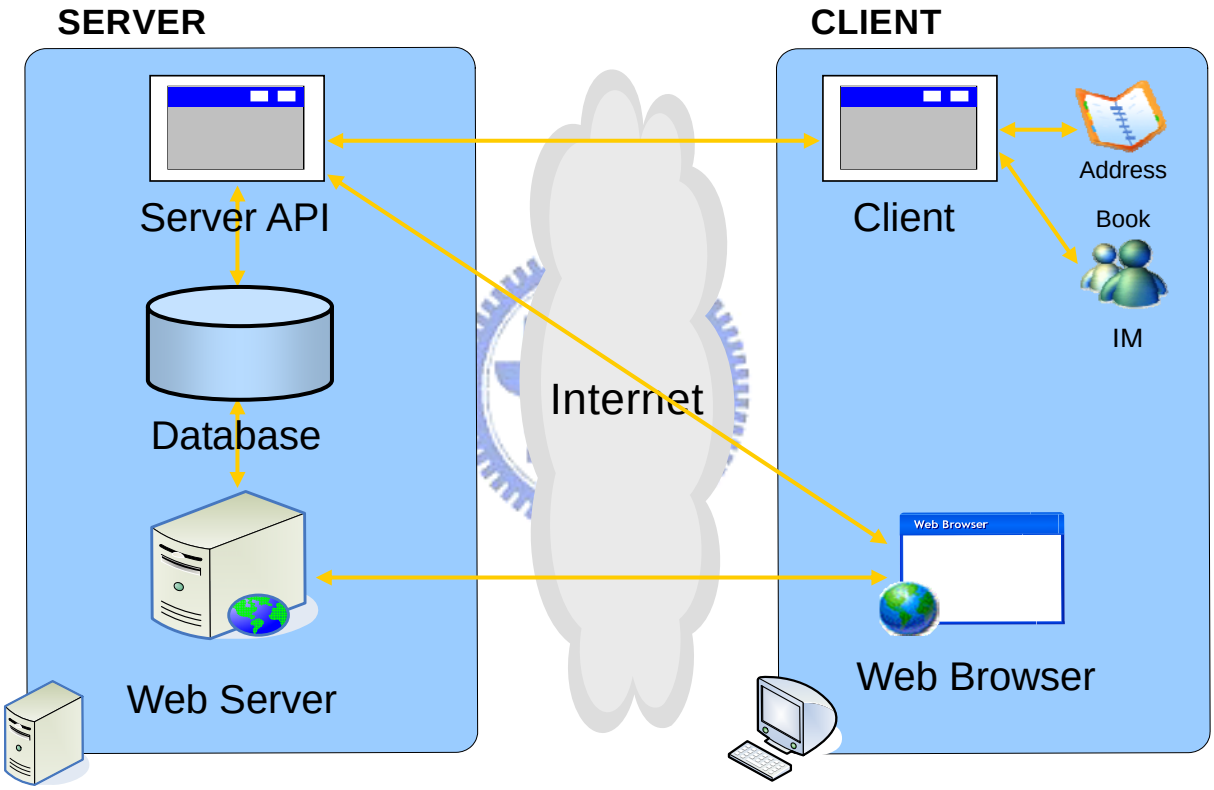


圖7 系統架構圖

伺服器端以一資料庫作為儲存使用者資訊、聯絡人資訊、聯絡人管理資訊以及聯絡人上線狀態之媒介；網頁伺服器則負責提供 Web 使用者介面。

伺服器包裝資料庫之低階存取動作，提供一組 API 提供客戶端程式存取資料庫內容。客戶端程式可透過此組 API 進行使用者身份認證、聯絡人管理、聯絡人狀態存取以及即時通訊之用。另外我們也將這組 API 設計為可供網頁瀏覽器存取，如此一來網頁瀏覽器與客戶端程式可共用一組 API，不必為了兩種介面而提供兩組不同的存取方式，簡化了程式的複雜度以及強化系統維護及擴充的簡便性。未來若納入本系統來開發其他系統，這組 API 也將做為其他系統與本系統溝通之介面。資料庫、網頁瀏覽器以及伺服器端 API 為本系統伺服器端之主要構成元件。

在客戶端方面，我們有一支客戶端程式負責與伺服器溝通。客戶端程式具有三種主要功能：與伺服器端溝通之網路連線功能、與本地端即時通訊程式溝通之功能，以及存取本地端聯絡人資料庫之功能。

客戶端程式啟動後，會先呼叫伺服器端 API，以使用者設定之帳號密碼進行與伺服器端認證之工作，完成登入手續。客戶端會定時比對伺服器端與客戶端之聯絡人資料庫，判斷是否需要更新。在需要更新的時候，再透過伺服器端 API 或本地端的聯絡人資料庫存取功能，去存取兩邊的資料，將兩端之聯絡人資料庫保持在同步的狀態。另外客戶端也會定時根據使用者的聯絡人的即時通訊帳號，去向客戶端的即時通訊程式查詢聯絡人的上線狀態，並同步伺服器端之上線狀態資訊。

當使用者透過網頁瀏覽器存取聯絡人時，瀏覽器也透過同一組 API 登入伺服器，以及取得聯絡人資料，此時使用者就可在 Web 介面上上看到最新的聯絡人資訊。當使用者選定要通訊的對象，系統就根據使用者的命令讓瀏覽器呼叫對應之客戶端即時通訊程式，並建立與聯絡人之通訊連線。

4.2 伺服器端之系統設計

4.2.1 伺服器端架構

我們根據4.1節所設計之架構來實作系統。底層之資料庫，我們採用MySQL，網頁介面則使用PHP來提供。我們的API採用HTTP介面，這組API也透過PHP來實作，client端程式、瀏覽器或第三方的系統，只要懂得HTTP協定就可以與我們的伺服器溝通。圖8為伺服器端實作之總體架構圖，可以看到同一組API可提供我們的客戶端程式存取，也提供了部分Web User Interface 使用。

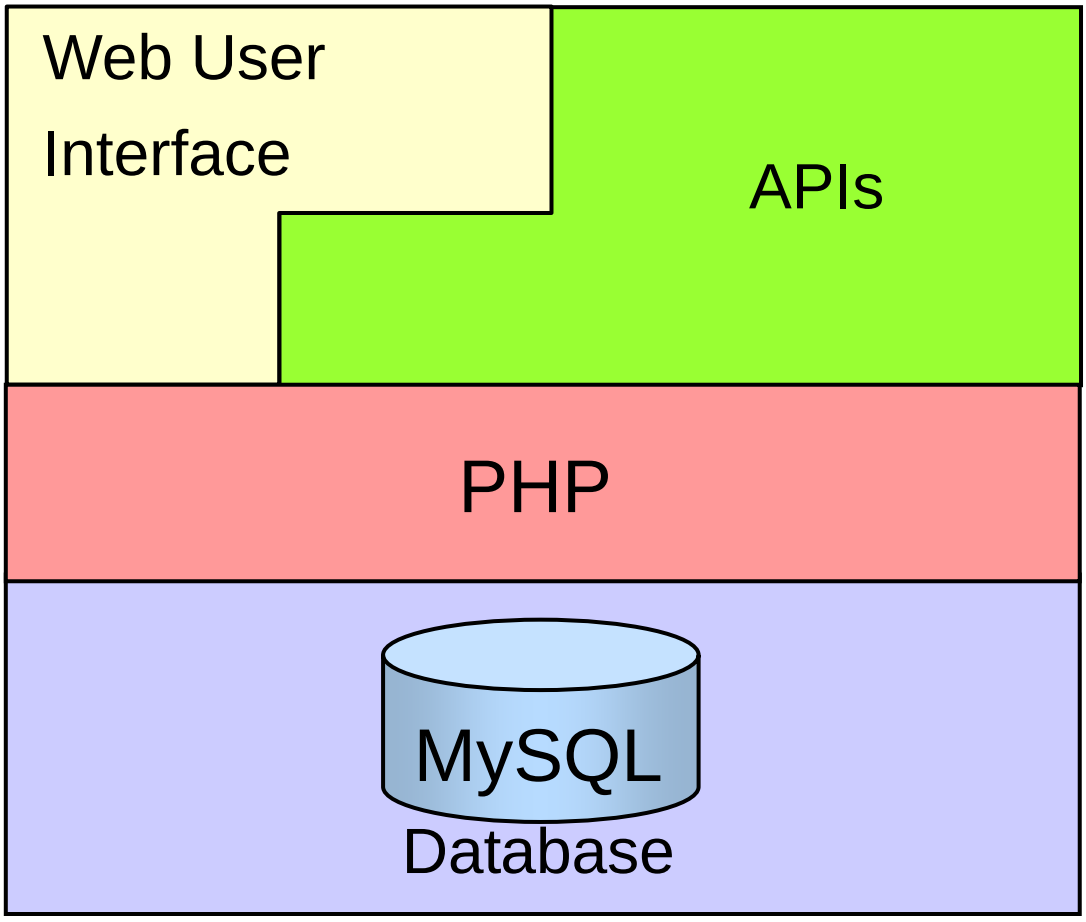


圖8 伺服器端架構圖

4.2.2 聯絡人管理概念

伺服器端對於聯絡人管理有兩個基本元素：聯絡人(Contacts)以及群組(Group)。各具有以下特性：

聯絡人

- **每個聯絡人以一個 Contact ID (CID)來識別。**
  CID 是一個 34 字元之字串，開頭為大寫'CS'，以避免在伺服器端與客戶端產生出相同之 CID；其餘 32 字元為不重複字串。CID 作為在資料庫中識別獨一無二聯絡人的鍵值(key)，也作為在伺服端與客戶端間區別一位聯絡人的基準。在新增一位聯絡人時，CID 會由伺服器產生。CID 一旦產生即不可改變。
- **聯絡人物件記錄了該聯絡人的通訊資料。**
  包括姓名、電話、地址、e-mail、即時通訊帳號等。
- **即時通訊的上線狀態也被視為聯絡人的屬性之一。**
- **每個聯絡人都有一個時間戳記(timestamp)，稱為 LAST_MODIFICATION_TIME。**
  紀錄該聯絡資料最後一次被修改的時間。除上線狀態外，對聯絡人屬性作任何修改時，系統會改變此時間戳記為伺服器當時的時間，而其初始值則為該聯絡人被產生的時間。此欄位作為同步比對之用，為決定哪一端的聯絡人資訊較新之依據。
- **系統允許對聯絡人進行新增、刪除、修改以及查詢之動作**
  這些動作都透過伺服器端 API 完成。

群組

- **群組為聯絡人管理之基本單位。**
- **如同聯絡人，群組也有一個識別代號，稱為 Group ID (GID)。**
  GID 是一 34 字元之字串，特性與 CID 相同，惟其開頭為大寫字母 'GS'。
- **群組可以包含兩種可能的元素：聯絡人以及子群組**
  子群組也是一個群組，但是一個群組並不能循環地包含它自己，這在透過 API 將群組加入另一個群組時會做檢查。群組可以被多個群組所包含，即一個群組可能同時為兩個群組之子群組。

- **群組也具備 LAST_MODIFICATION_TIME 時間戳記**
  初始值為建立該群組之時間，在修改群組名稱以及群組內容時改變，作為同步比對之依據。

- **系統也允許對群組進行新增、刪除、修改以及查詢之動作，除此之外還支援加入群組元素、刪除群組元素兩個動作。**
  如同聯絡人，這些動作在伺服器端也透過 API 來完成。

4.2.3 資料庫結構

在伺服器端，我們以資料庫作為儲存使用者資訊、聯絡人資訊、聯絡人管理資訊以及上線狀態的媒介。考量到國際化的方便，資料庫中所有文字資料都以 Unicode UTF-8 格式儲存，而時間資料則以格林威治標準時間(GMT)為準。如此不同語系不同時區的使用者，可有一個共同的基準來進行資料的存取以及同步。

我們依據需求規劃出表格結構：

為使用者資訊表格 USER_PROFILES，重要欄位說明如下：UID 是一 33 字元之使用者編號，在系統新增使用者時自動產生並指派。用以識別每一位使用者，以及從其他表格中取出關連於該使用者的資訊；USERNAME 與 PASSWORD 是使用者登入之帳號與密碼，其中 PASSWORD 是以單向加密儲存於表格中；PREFERENCE 則儲存使用者的偏好設定。

表1 使用者資訊

User_Profiles		
Field	Data type	Description
UID	CHAR (33)	使用者編號
USERNAME	CHAR (33)	使用者帳號
PASSWOED	CHAR (41)	使用者密碼
NICKNAME	CHAR (128)	使用者暱稱
EMAIL	CHAR (255)	使用者電子郵件帳號
PREFERENCE	CHAR (255)	使用者偏好設定

錯誤! 找不到參照來源。 為使用者活動狀態表格 USER_ACTIVITIES，重要欄位說明如下：SESSION_ID 是 PHP 之登入階段(session)機制產生之 32 字元之使用者編號，在使用者登入時產生，登出或逾時時消滅。UID 是使用者的編號，而時間戳記 LAST_ACTIVITY_TIME 記錄了使用者上次活動的時間。當使用者對伺服器做任何請求、或是執行 keep alive 動作時，此時間戳記會更新為當時的伺服器時間。系統用以判斷某個使用者登入階段是否逾時，以進行必要的處理。

表2 使用者活動狀態

User_Activities		
Field	Data type	Description
SESSION_ID	CHAR (32)	登入階段編號
UID	CHAR (33)	使用者編號
LAST_ACTIVITY_TIME	TIMESTAMP	使用者上次活動的時間

為聯絡人資訊表格，重要欄位說明如下：CID即4.2.2描述之Contact ID；UID為

之使用者之編號，用以區別該筆聯絡人資料是屬與哪一位使用者所有。GIVEN_NAME、SURNAME以至於ADDRESS_STREET為該聯絡人之個人資料。其中三種電話號碼(TELEPHONE_NUMBER、CELLPHONE_NUMBER、FAX_NUMBER)，在使用者要求與聯絡人電話聯繫時，可成為撥出的目標。EMAIL是聯絡人之電子郵件位址，使用者要求與對方電子郵件聯繫時，會使用這個位址。MSN與SKYPE為目前系統支援之兩種即時通訊系統帳號，客戶端會根據這兩個資料來獲取該帳號的上線狀況。LAST_MODIFICATION_TIME則為4.2.2描述之聯絡人最後修改時間，除了變更MSN_STATUS或SKYPE_STATUS外，變動其餘欄位皆會將LAST_MODIFICATION_TIME改變為目前時間。

MSN_STATUS與SYKPE_STATUS記錄了該聯絡人之上線狀態，這兩個欄位可能的內容根據MSN Messenger與Skype的官方設定，列於**錯誤! 找不到參照來源。**，在聯絡人上線狀態不明時(如使用者未登入)會設為MSN_UNKNOWN、SKYPE_UNKNOWN。

表3 聯絡人資訊

Contacts		
Field	Data type	Description
CID	CHAR (34)	聯絡人編號(Contact ID, CID)
UID	CHAR (33)	使用者編號
GIVEN_NAME	CHAR (32)	聯絡人名
SURNAME	CHAR (32)	聯絡人姓
TELEPHONE_NUMBER	CHAR (32)	一般電話號碼
CELLPHONE_NUMBER	CHAR (32)	行動電話號碼
FAX_NUMBER	CHAR (32)	傳真號碼
ADDRESS_COUNTRY	CHAR (128)	聯絡地址：國家
ADDRESS_STATE_OR_PROVINCE	CHAR (128)	聯絡地址：洲/省、台灣則視為縣市

Contacts		
Field	Data type	Description
ADDRESS_CITY	CHAR (128)	聯絡地址：城市，台灣則視為鄉鎮市區
ADDRESS_STREET	CHAR (255)	聯絡地址：街道地址
EMAIL	CHAR (255)	電子郵件帳號
MSN	CHAR (255)	MSN Messenger 帳號
SKYPE	CHAR (255)	Skype 帳號
LAST_MODIFICATION_TIME	TIMESTAMP	該聯絡人資訊最後被修改的時間
MSN_STATUS	CHAR(16)	聯絡人的 MSN Messenger 上線狀態
SKYPE_STATUS	CHAR(16)	聯絡人的 Skype 上線狀態

為群組資訊表格，重要欄位說明如下：GID是4.2.2描述之Group ID；UID為

之使用者之編號，用以區別該筆群組資料是屬與哪一位使用者所有。NAME是群組的名稱。LAST_MODIFICATION_TIME是4.2.2描述之最後修改時間戳記，在改變群組名稱，或群組內含元素時，此時間戳記會被改變為伺服器當時之時間。

表4 群組資訊

Groups		
Field	Data type	Description
GID	CHAR (34)	群組編號 (Group ID, GID)
UID	CHAR (33)	使用者編號
NAME	CHAR (128)	群組名稱
LAST_MODIFICATION_TIME	TIMESTAMP	群組最後修改時間

為群組內容表格，每一筆資料代表一個群組的元素，重要欄位說明如下：UID為

之使用者之編號，用以區別該筆群組內容資料是屬與哪一位使用者所有。GID 為某個群組之 Group ID；ID 則是 GID 所內含之某個群組元素 ID 值，可以是 GID 也可以是 UID。舉例而言，若某個使用者 UIDi 群組 GIDA 中有兩個子群組 GIDB 以及 GIDC，還有一個聯絡人 CID1，則在表格中會以三筆紀錄表示：(UIDi,GIDA,GIDB)、(UIDi,GIDA,GIDC)、(UIDi,GIDA,CID1)。

表5 群組內容

Group_Entries		
Field	Data type	Description
UID	CHAR (33)	使用者編號
GID	CHAR (34)	群組編號(Group ID, GID)
ID	CHAR (34)	群組元素編號

4.2.4 伺服器端 API

我們將伺服器端提供的功能包裝為一組 API，供客戶端程式、網頁瀏覽器以及第三方系統存取。客戶端以 HTTP 呼叫這些 API，伺服器提供相應的服務，API 的回應以及輸出資料則為 XML 格式，以方便呼叫端處理回傳的資料。文字編碼則考量國際化的需求，為 UTF-8。

API 回傳的訊息有兩種，若失敗會回傳：

```
<result type="error">                          //result type 爲 error
  <errtype>SOME_ERROR_TYPE</errtype>           //錯誤類型
  <message>SOME_ERROR_MESSAGE</message>        //錯誤訊息
</result>
```

成功則有兩種可能，若是要求進行某個動作（如要求登入），成功會回傳：

```
<result type="success">                        //result type 爲 success
  <message>SOME_SUCCESS_MESSAGE</message>  //某動作成功的訊息
</result>
```

若是向伺服器要求資料（如要求聯絡人清單），則是下列的格式

```
<result type="TYPE_OF_DATA">           //result type 爲資料的類型
  <data1>                                    //以 XML 表示的資料列表
    <data1-1>data one of one</data1-1>       //資料內的資料
    <data1-2>data two of two</data1-2>
    ...
  </data1>
  <data2>
    <data2-1>data one of one</data2-1>
    <data2-2>data two of two</data2-2>
    ...
  </data2>
  ...
</result>
```

各 API 以功能區別，描述如下：

使用者認證、登入階段管理

- **api_login**
  參數：username, password

客戶端登入，傳入使用者之帳號密碼，系統會根據帳號密碼來確認使用者的身份，若成功，會啟動一登入階段(session)，並回傳成功訊息。必須先以 api_login 登入，才可呼叫其餘的 API，否則系統會拒絕用戶端的 API 呼叫，並回傳 AUTH_NOT_LOGIN_YET 錯誤。

可能回傳的失敗類型：

AUTH_LOGIN_ERROR：登入錯誤

- **api_logout**
  參數：無

  客戶端登出，呼叫時系統會結束之前呼叫 api_login 所建立之登入階段。若成功會回傳成功訊息。

  可能回傳的失敗類型：

  AUTH_NOT_LOGIN_YET：尚未登入

- **api_keep_alive**
  參數：無

  維護目前進行中的登入階段，避免因爲逾時而被系統判定客戶端已離線。不過登入後呼叫任何 API 存取時，伺服器都會隱含地進行這個動作，故此 API 是供客戶端已知自己一段時間不會與伺服器端溝通，而又必須維護登入狀態時使用。

  可能回傳的失敗類型：

  AUTH_NOT_LOGIN_YET：尚未登入

- **api_get_activities**
  參數：無

  回傳某使用者目前進行中的所有登入階段內容。

  成功時回傳的資料格式如下：

```
<result type="activities">
  <activity>
    <sessionid>[ID OF SESSION1]</sessionid>
  </activity>
  <activity>
    <sessionid>[ID OF SESSION2]</sessionid>
```

```
  </activity>
  ...
</result>
```

可能回傳的失敗類型：

AUTH_NOT_LOGIN_YET：尚未登入

使用者認證、登入階段管理

- **api_get_profile**

  參數：無

  供客戶端取得使用者的個人資訊，包含使用者名稱、電子郵件帳戶、偏好設定等。密碼因為是以單向加密儲存在資料庫，故不回傳給客戶端，若客戶端要更改密碼，可以使用 api_set_profile。

  成功時回傳的資料格式如下：

```
<result type="profile">
  <username>[USERNAME OF USER]</username>
  <nickname>[NICKNAME OF USER]</nickname>
  <email>[EAMIL OF USER]</email >
  <preference>[PREFERENCE OF USER]</preference>
</result>
```

  可能回傳的失敗類型：

  AUTH_NOT_LOGIN_YET：尚未登入

- **api_set_profile**

  參數：nickname、password、email、preference

  設定使用者個人資料，包括密碼，暱稱、電子郵件位址以及偏好設定。

  可能回傳的失敗類型：

  AUTH_NOT_LOGIN_YET：尚未登入

聯絡人與群組管理

- **api_add_contact**
  參數：cid, given_name, surname, telephone_number, cellphone_number, fax_number, address_state_or_province, address_country, address_city, address_street, email, msn, skype

  根據傳入的資料在伺服器端資料庫裡新增一個聯絡人，如果有傳入 cid 欄位，就會將新聯絡人的 Contact ID 設定為傳入的 cid，若沒有則由伺服器自動產生。此動作會將新聯絡人之 LAST_MODIFICATION_TIME 設為產生此聯絡人時之時間。

  可能回傳的失敗類型：

  AUTH_NOT_LOGIN_YET：尚未登入
  CONTACT_EMPTY_INFO：傳入的聯絡人資料爲空

- **api_delete_contact**
  參數：cid

  根據傳入的 cid，從伺服器端資料庫裡刪除指定 Contact ID 的聯絡人。同時也會砍除 GROUP_ENTRIES 裡包含此聯絡人 CID 之記錄。

  可能回傳的失敗類型：

  AUTH_NOT_LOGIN_YET：尚未登入
  CONTACT_EMPTY_CID：傳入的 CID 爲空

- **api_modify_contact**
  參數：cid, given_name, surname, telephone_number, cellphone_number, fax_number, address_state_or_province, address_country, address_city, address_street, email, msn, skype

  根據傳入的資料修改伺服器端資料庫裡由 cid 指定的聯絡人資料。此動作會將聯絡人之 LAST_MODIFICATION_TIME 設為修改聯絡人資料時之時間。

  可能回傳的失敗類型：

  AUTH_NOT_LOGIN_YET：尚未登入
  CONTACT_EMPTY_CID：傳入的 CID 爲空

- **api_get_contacts**

  參數：cids

  參數 cids 為聯絡人 Contact ID 之序列，各 CID 間以逗號分隔(cid1,cid2,cid3,...)。伺服器端會傳回 cids 指定之複數聯絡人資料，若未指定 cids，則會回傳屬於該使用者之所有聯絡人資料。

  成功時回傳的資料格式如下：

```
<result type="contact_list">
  <contact>
    <cid>[CID OF CONTACT]</cid>
    <surname>[SURNAME OF CONTACT]</surname>
    <given_name>[GIVEN NAME OF CONTACT]</given_name>
    <telephone_number>[TELEPHONE # OF CONTACT]</telephone_number>
    <cellphone_number>[CELLPHONE # OF CONTACT]</cellphone_number>
    <fax_number>[FAX # OF CONTACT]</fax_number>
    <address_state_or_province>
      [ADDRESS - STATE OR PROVINCE OF CONTACT]
    </address_state_or_province>
    <address_country>
      [ADDRESS - COUNTRT OF CONTACT]
    </address_country>
    <address_city>
      [ADDRESS – CITY OF CONTACT]
    </address_city>
    <address_street>
      [ADDRESS – STREET OF CONTACT]
    </address_street>
    <email>[E-MAIL OF CONTACT]</email>
    <msn>[MSN MESSENGER SIGNIN NAME OF CONTACT]</msn>
    <Skype>[SKYPE SIGNIN NAME OF CONTACT]</skype>
    <last_modification_time>
      [LAST MODIFICATION TIME OF CONTACT RECORD]
    </last_modification_time>
    <msn_status>[MSN MESSENGER STATUS OF CONTACT]</msn_status>
    <skype_status>[SKYPE STATUS OF CONTACT]</skype_status>
  </contact>

  <contact> ... </contact>
```

```
  <contact> ... </contact>
   ...
</result>
```

可能回傳的失敗類型：

AUTH_NOT_LOGIN_YET：尚未登入

- **api_add_group**
  參數：gid, name

  根據傳入的群組名稱在伺服器端資料庫裡新增一個群組，如果有傳入 gid 欄位，就會將新群組的 Group ID 設定為傳入的 gid，若沒有則由伺服器自動產生。此動作會將新群組之 LAST_MODIFICATION_TIME 設為產生此群組時之時間。

  可能回傳的失敗類型：

  AUTH_NOT_LOGIN_YET：尚未登入
  GROUP_EMPTY_NAME：傳入的群組名稱爲空

- **api_delete_group**
  參數：gid

  根據傳入的 gid，從伺服器端資料庫裡刪除指定 Croup ID 之群組。同時也會砍除 GROUP_ENTRIES 裡關於此群組之記錄，包括此群組所包含之元素，以及包含此群組為元素之群組。

  可能回傳的失敗類型：

  AUTH_NOT_LOGIN_YET：尚未登入
  CONTACT_EMPTY_GID：傳入的 GID 爲空

- **api_modify_group**
  參數：gid, name

  根據傳入的資料修改伺服器端資料庫裡由 gid 指定之群組資料。此動作會將該群組之 LAST_MODIFICATION_TIME 設爲修改群組資料時之時間。

  可能回傳的失敗類型：

  AUTH_NOT_LOGIN_YET：尚未登入
  CONTACT_EMPTY_GID：傳入的 GID 爲空
  GROUP_EMPTY_NAME：欲修改之群組名稱爲空

- **api_get_groups**
  參數：gids

  參數 gids 爲群組 Group ID 之序列，各 GID 間以逗號分隔(gid1,gid2,gid3,...)。伺服器端會傳回 gids 指定之複數群組資料，若未指定 gids，則會回傳屬於該使用者之所有群組資料。

  成功時回傳的資料格式如下：

```
<result type="group_list">
  <group>
    <gid>[GID OF GROUP]</gid>
    <name>[NAME OF GROUP]</name>
    <last_modification_time>
      [LAST MODIFICATION TIME OF GROUP RECORD]
    </last_modification_time>
    <entries>
      <entry>[ENTRY 1 OF GROUP]</entry>
      <entry>[ENTRY 2 OF GROUP]</entry>
       ...
    </entries>
  </group>
  <group>
   ...
  </group>
   ...
</result>
```

  可能回傳的失敗類型：

  AUTH_NOT_LOGIN_YET：尚未登入

- **api_add_entry_to_group**
  參數：gid, id

  將 id 指定之元素加入由 gid 指定之群組為群組元素，id 可為 cid 或 gid。此 API 會檢查母群組以及元素是否將形成循環包含，若形成循環包含則會拒絕要求並回傳錯誤類型。加入一個群組元素將會使母群組 gid 之 LAST_MODIFICATION_TIME 改變為加入群組元素之時間，但不會改變群組元素 id 之 LAST_MODIFICATION_TIME。

  可能回傳的失敗類型：

  AUTH_NOT_LOGIN_YET：尚未登入
  GROUP_EMPTY_GID：傳入的 GID 爲空
  GROUP_EMPTY_ID：傳入之群組元素 ID 爲空
  GROUP_PARENT_UNEXISTED：指定之母群組不存在
  GROUP_INVALID_ENTRY_ID：傳入之群組元素 ID 不正確
  GROUP_ENTRY_UNEXISTED：指定之群組元素不存在
  GROUP_CYCLIC_ENTRY：加入群組元素將形成循環包含

- **api_delete_entry_from_group**
  參數：gid, id

  將群組元素 id 自 gid 指定之群組中刪除，id 可為 cid 或 gid。刪除群組元素並不會將該聯絡人或群組自資料庫中刪除。刪除一個群組元素將會使母群組 gid 之 LAST_MODIFICATION_TIME 改變為刪除群組元素之時間，但不會改變群組元素 id 之 LAST_MODIFICATION_TIME。

  可能回傳的失敗類型：

  AUTH_NOT_LOGIN_YET：尚未登入
  GROUP_EMPTY_GID：傳入的 GID 爲空
  GROUP_EMPTY_ID：傳入之群組元素 ID 爲空
  GROUP_PARENT_UNEXISTED：指定之群組元素不存在

- **api_clear_all**

  參數：無

  清空目前使用者所擁有之所有聯絡人與群組內容。

  可能回傳的失敗類型：

  AUTH_NOT_LOGIN_YET：尚未登入

聯絡人上線狀態更新

- **api_set_presence.php**

  參數：type, cids, status

  設定聯絡人之即時通訊上線狀態。type 指定上線狀態的類型, 如 msn, Skype,…cids 指定要設定狀態的聯絡人 CID 序列，CID 間以逗號分隔(cid1,cid2,...)status 指定對應 cid1,cid2,...的上線狀態，也以逗號分隔(status1,status2,...)。接受的參數值視 type 類型而不同，msn 類型與 skype 類型可接受之狀態值根據 MSN 與 Skype 之官方設定訂定，列於**錯誤! 找不到參照來源。**。若 cids 與 status 包含之序列長度不同，會以 cids 為主，若 cids 數量比 status 少，則剩下的會設為 MSN_UNKNOWN、SKYPE_UNKNOWN。若只有傳入 type 而 cids 為空，則會全部設成 status1，若沒有 status1, 則會設成 MSN_UNKNOWN、SKYPE_UNKNOWN。

  可能回傳的失敗類型：

  AUTH_NOT_LOGIN_YET：尚未登入

  PRESENCE_UNKNOWN_TYPE：系統不認識指定之上線狀態類型

表6 聯絡人上線狀態可接受之參數

msn 類型之狀態	skype 類型之狀態
MSN_ONLINE	SKYPE_ONLINE
MSN_OFFLINE	SKYPE_OFFLINE
MSN_AWAY	SKYPE_AWAY
MSN_IDLE	SKYPE_NA
MSN_BE_RIGHT_BACK	SKYPE_DND
MSN_BUSY	SKYPE_SKYPEME
MSN_ON_THE_PHONE	SKYPE_SKYPEOUT
MSN_OUT_TO_LUNCH	SKYPE_UNKNOWN*
MSN_UNKNOWN*	

* 在聯絡人上線狀態不明時，上線狀態爲 MSN_UNKNOWN、SKYPE_UNKNOWN

伺服器端時間存取

- **api_get_server_time.php**
  參數：無

  取得目前伺服器端之時間，可作為同步比對之用。傳回之時間時區皆為 GMT，格式為 ISO 8601：YYYY-MM-DD HH:NN:SS。

  可能回傳的失敗類型：

  AUTH_NOT_LOGIN_YET：尚未登入

- **api_get_last_modification_time.php**
  參數：無

  取得目前伺服器端屬於目前使用者之群組與聯絡人最後修改時間，可做為客戶端判定是否更新資料之依據。傳回之時間時區皆為 GMT，格式為 ISO 8601：YYYY-MM-DD HH:NN:SS。

  可能回傳的失敗類型：

  AUTH_NOT_LOGIN_YET：尚未登入

除上述之 API 傳回錯誤類型外，若伺服器端在執行 API 之過程中出現錯誤，也可能回傳一些錯誤類型：

DB_CONNECTION_ERROR：資料庫連線錯誤
DB_SELECTION_ERROR：資料庫選擇錯誤
DB_QUERY_ERROR：資料庫查詢錯誤

4.2.5 伺服器面對客戶端要求之處理過程

以下就以幾個實例來說明上述之伺服器端結構，在實際收到使用者要求之詳細運作過程。

4.2.5.1 聯絡人上線狀態之維護

現在有一客戶端程式登入伺服器端，伺服器會在 USER_ACTIVITIES 建立一筆新紀錄，記錄目前客戶端之 UID、SESSION ID 以及登入時的時間。登入後後客戶端可使用 api_set_presence 設定客戶端擁有之聯絡人上線狀態。

使用者欲存取聯絡人資料，用瀏覽器從 Web 介面登入系統。Web 介面使用 api_get_contacts 要求聯絡人清單，並在 Web 介面顯示出來。

若客戶端程式登出系統，伺服器端會清除 USER_ACRIVITIES 中，該次登入所建立之登入階段記錄，並檢查是否還有客戶端在線上，若已無其他登入的客戶端，則將該使用者所屬之聯絡人狀態設定為 MSN/SKYPE_UNKNOWN。

但若客戶端不正常離線（如網路斷線），則該筆記錄在無法在登出時被清除，伺服器也無法主動檢查資料庫中是否有已經逾期的登入階段（因為 Web 伺服器乃是一被動之伺服端架構），所以需要一個方法讓伺服器在接受客戶端要求時，能順便進行清理逾期登入階段之工作。

因為登出程序涉及的有使用者活動狀態，以及聯絡人上線狀態的變動，而涉及這兩種資訊的取得的 API 有 api_user_login、api_get_activities 與 api_get_contacts 三種。伺服器端會在執行這三個 API 時，隱含地檢查 USER_ACTIVITIES 中是否有逾期的登入階段記錄，若有，則為該登入階段執行必要的登出程序（清除該記錄，重設聯絡人狀態）。如此，雖然資料庫中的資料不是時時保持新鮮，卻可以確保在使用者要求時可以獲得最新的狀況。

4.2.5.2 即時通訊

若使用者透過 Web 介面要求與某個聯絡人通訊，伺服器會根據使用者要求的方式，從伺服器端資料庫取出該位聯絡人的聯繫資料。而要怎麼呼叫客戶端的對應程式呢？系統會產生一組 URI(Universal Resource Indicator)，瀏覽器收到這組 URI，會叫起對應的客戶程式來處理。

URI 格式會有兩種情況：如果客戶端的即時通訊程式本身有處理 URI 的能力，則會直接回傳該程式所認識之 URI 協定（如 skype:、mailto:…等）；而如果客戶端的即時通訊程式無法處理 URI，則會由我們的客戶端程式代為呼叫。此情況所回傳之 URI 協定格式為 contacto:[MESSENGER ACCOUNT]?[METHOD]，例如客戶端要求與 somebody@msn.com 以 MSN Messenger 進行文字交談，而 MSN Messenger 沒有處理 URI 的能力，則伺服端會回傳：contactto:somebody@msn.com?msn_text 這樣的 URI，瀏覽器收到以後，會呼叫我們的客戶端程式，客戶端程式分析此 URI，再叫起 MSN Messenger，並打開與 somebody@msn.com 的文字交談視窗。

4.2.5.3 聯絡人管理

聯絡人管理所涉及的資料庫表格有三個：CONTACTS、GROUPS、GROUP_ENTRIES。以下以刪除某個聯絡人、與刪除某個群組的程序來說明此三個表格的關聯互動情況。

若伺服端收到刪除某個聯絡人CID_C的要求(api_delete_contact)，伺服器端會先從GROUP_ENTRIES找出所有ID欄位為CID_C之紀錄，並將之刪除，即刪除所有包含此聯絡人之群組裡，代表此聯絡人之元素。接下來伺服器端會在CONTACTS表格裡找到CID_C，並刪除該記錄，即刪去此聯絡人。至此，一個聯絡人已經成功從伺服端被刪除，

圖 9 顯示CID_C

CONTACTS

CID	Name
CID_B	Contact B
CID_C	Contact C
CID_D	Contact D

GROUP_ENTRIES

GID	ID
GID_X	GID_F
GID_Y	CID_C
GID_Z	CID_C

a.

b.

c.

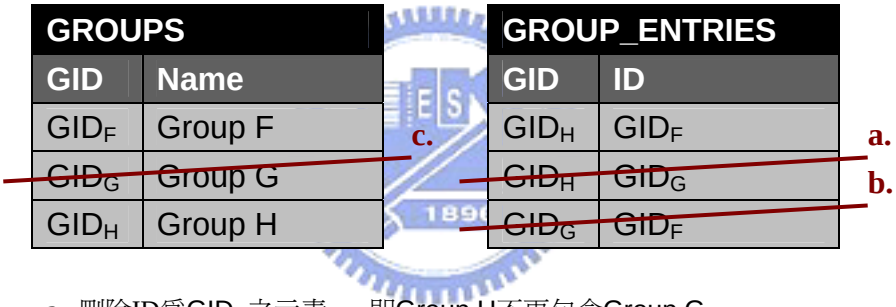
a. b. 刪除ID爲CID_C之元素 – 即GID_Y、GID_Z不再包含CID_C

c. 刪除CID爲CID_C之元素 – 即將CIDc從資料庫中刪除

圖9 從資料庫中刪除Contact C (CID_C)之過程

若伺服端收到刪除某個群組GIDG的要求(api_delete_group)，伺服器端會先從GROUP_ENTRIES找出所有ID欄位為GIDG之紀錄，並將之刪除，即刪除所有包含此群組之母群組裡，代表此群組之元素。接下來再刪除GROUP_ENTRIES裡，GID欄位為GIDG之紀錄，即刪除所有此群組的元素記錄（注意，並不是從資料庫裡刪除這些元素，只是刪除群組GIDG包含該元素之記錄）。接下來伺服器端會在GROUPS表格裡找到該GIDG，並刪除該記錄，即刪去此群組。此為一個群組從伺服端被刪除的過程，

圖 10 顯示群組GIDG被刪除的過程。



a. 刪除ID爲GID_G之元素 – 即Group H不再包含Group G

b. 刪除GID爲GID_G之元素 – 即Group G不再包含任何元素

c. 刪除ID爲GID_G之元素 – 即將Group G從資料庫刪除

圖10 從資料庫中刪除Group G (GID_G)之過程

4.3 客戶端之系統設計

4.3.1 客戶端架構

客戶端負責作為本地端與伺服器端溝通的橋樑，需要具備的能力有：本地端聯絡人資料庫的存取、伺服器端聯絡人資料庫的存取、即時通訊上線狀態的感知、來自伺服端 URI 命令的處理，以及呼叫 URI 所指定的客戶端即時通訊程式的能力。

在聯絡人資料的方面，本系統目前實作的目標是 Windows 作業系統所內建之 Windows Address Book（WAB，中文版 Windows 系統稱為「通訊錄」)。而我們希望未來能夠方便擴充支援其他的聯絡人資料庫系統，所以我們將聯絡人資料存取所需的動作，抽取出來形成一抽象層 Contact Management，而伺服器端的聯絡人資料庫存取與本地端的資料庫存取動作類似，也納入此架構之下。在實作中我們繼承此一介面，實作了 WAB 以及 Server 端聯絡人的存取。

在聯絡人上線狀態的感知方面，系統目前實作支援的有 MSN Messenger 以及 Skype。同樣地，為了未來的擴充考量，也設計一個抽象介面。在此介面下實作 MSN Messenger 以及 Skype 的狀態感知功能。

在即時通訊程式的控制方面，雖然實作中只有 MSN Messenger 需要由我們的客戶端來控制，但也實作了抽象介面 IM Controlling，以滿足未來支援其他的即時通訊程式之需求。

再加上URI的處理，客戶端用一個Application Controller來控制協調上述模組間的協同運作，再提供使用者介面讓使用者控制程式之行為。形成客戶端之系統架構，如圖 11。

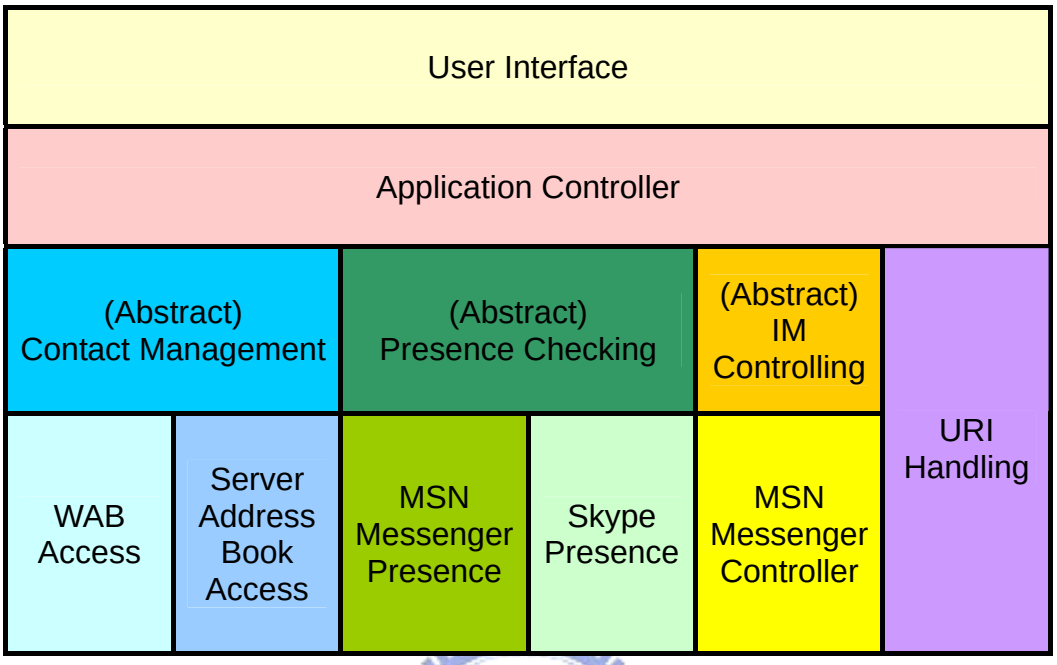


圖11 客戶端架構圖

4.3.2 與即時通訊系統之互動

客戶端程式與即時通訊系統之互動，有兩種情形：獲取某個聯絡人的上線狀態，以及呼叫某個即時通訊系統之客戶端程式。在實作裡我們支援了兩種即時通訊系統，MSN Messenger 以及 Skype。

MSN Messenger 提供了官方的 API，稱為 Messenger Type Library，MTL 為微軟之 Windows Messaging Service 提供之 COM 介面，Windows 內建的 Windows Messenger，以及 MSN 的 MSN Messenger 都是使用同樣的核心，在使用 COM 函式庫的時候，只要選擇具現(instance)不同的介面，就可以控制不同的程式(IMessenger: Windows Messenger、IMessenger3: MSN Messenger)。Skype 也同樣使用了 COM 介面來提供外部程式存取 Skype 客戶端程式的 API，稱為 Skype4Com。我們的客戶端程式使用這兩個 API 來實作對於即時通訊程式的互動功能。

在上線狀態方面，我們的抽象介面稱為IMChecker（請參考

)。此介面提供了一個方法，稱為 getStatus()，傳入要獲取的聯絡人即時通訊帳號，此方法會回傳該聯絡人之狀態，狀態型別為 IMStatus。而各種可能的狀態，則由繼承此介面之類別，視其所實作之即時通訊系統來自行定義。我們包裝 MSN Messenger 以及 Skype API 之底層動作，實作的兩個類別稱為 MSNChecker 以及 SkypeChecker。

表7 IMChecker介面之方法列表

Name	Parameters	Retuen	Description
getSTatus	user: String	IMStatus	取得即時通訊代號為 user 之聯絡人上線狀態 IMStatus

在即時通訊程式的呼叫方面，我們的抽象介面稱為 IMController（請參考**錯誤! 找不到參照來源。**），提供的方法有 openText()、openVoice()、openVideo()與 openPhone()，呼叫時傳入通訊目標之帳號，即可建立對應之對話連線。實作的類別為 MSNController，Application Controller 收到瀏覽器傳來之 URI，可使用 MSNController 類別打開 MSN Messenger 指定類型的交談視窗。

表8 IMController 介面之方法列表

Name	Parameters	Retuen	Description
openText	target: String	-	與即時通訊代號為 target 之聯絡人建立文字交談連線
openVoice	target: String	-	與即時通訊代號為 target 之聯絡人建立語音交談連線
openVideo	target: String	-	與即時通訊代號為 target 之聯絡人建立視訊交談連線
openPhone	target: String	-	撥出 target 之聯絡人電話號碼

4.3.3 伺服器端與本地端聯絡人資料庫之存取

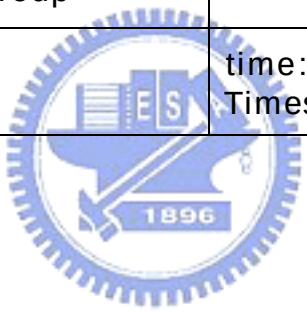
聯絡人資料庫之存取，我們的抽象介面稱為 ContactManager（請參考**錯誤! 找不到參照來源。**）。

表9 ContactManager 介面之方法列表

Name	Parameter	Return	Description
clear	-	-	清除目前聯絡人資料庫中所有資料
getAllContacts	-	contactList: ContactList	取回目前聯絡人資料庫中所有聯絡人
addContact	contact: Contact	-	將聯絡人 contact 加入聯絡人資料庫
delContact	contact: Contact	-	將聯絡人 contact 自聯絡人資料庫刪除
modifyContact	contact: oldContact contact: newContact	-	修改資料庫中已存在之聯絡人 oldContact 資料為 newContact
getAllGroups	-	groupList: GroupList	取回目前聯絡人資料庫中所有群組
addGroup	group: Group	-	將群組 group 加入聯絡人資料庫
delGroup	group: Group	-	將群組 group 自聯絡人資料庫刪除
modifyGroup	group: oldGroup group: newGroup	-	修改資料庫中已存在之群組 oldGroup 資料為 newGroup

Name	Parameter	Return	Description
addEntryToGroup	contact: Contact parentGroup: Group	-	將聯絡人 contact 加入群組 parentGroup 為群組元素
addEntryToGroup	group: Group parentGroup: Group	-	將聯絡人 contact 自群組 parentGroup 中刪除
deleteEntryFromGroup	contact: Contact parentGroup: Group	-	將群組 group 加入群組 parentGroup 為群組元素
deleteEntryFromGroup	group: Group parentGroup: Group	-	將群組 group 自群組 parentGroup 中刪除
getTime	-	time: Timestamp	取得目前聯絡人資料庫之時間

承上頁
續下頁



4.3.3.1 客戶端聯絡人資料庫

本地端聯絡人資料庫，支援的系統為Windows內建之Windows Address Book(WAB, wab.exe, 圖 6)。Windows Address Book提供一API稱為WAB API，可供第三方程式存取WAB之聯絡人資料庫。微軟本身亦利用這個API讓該公司旗下產品Outlook以及Outlook Express可使用WAB作為電子郵件通訊錄。此程式支援兩種聯絡人管理方案：資料夾以及群組。但資料夾屬於該程式自己的實作，並非API所提供的功能，也無法透過API來存取到資料夾結構，故在實作中我們利用其群組功能來讓客戶端與伺服器端之聯絡人管理結構同步。繼承ContactManager，包裝WAB API所實作之類別稱為WABContactManager。

在4.2.2提到，系統利用CID來作為在客戶端與伺服端間，識別一個聯絡人的方法。但是在客戶端要怎麼讓聯絡人與CID對應呢？WAB API提供了一個機制稱為Named Properties，可以讓第三方自訂WAB中聯絡人的屬性。我們的客戶端程式即利用Named Properties機制來將CID設定為聯絡人的屬性，即可將CID與聯絡人一一對應。

但是在本地端 WAB 程式裡，新增的聯絡人並不具此屬性。所以在取得本地端之聯絡人列表時，WABContactManager 若發現一個聯絡人不具此屬性，會自動產生一個 CID，但與伺服器端不同的是，WABContactManager 端產生之 CID，開頭是'CC'，以避免在伺服器端與客戶端間發生 CID 重複的情形。另外，在本地端產生之 CID，其產生的 CID 包括本地端獨一無二的 MAC Address，以避免同一位使用者在不同電腦使用多個客戶端，可能產生出相同 CID 的情形。同樣地，GID 也是透過類似的機制產生，其開頭為 'GC'。

4.3.3.2 伺服器端聯絡人資料庫

同樣地，我們也將4.2.4所描述之伺服器端API，以及相關的HTTP網路通訊，包裝為伺服器端之聯絡人管理類別，ServerContactManager。不同於WABContactManager的地方是，伺服器端有聯絡人上線狀態之屬性，所以ServerContactManager額外實作了關於聯絡人上線資訊之方法。

4.3.4 聯絡人資料庫同步

每隔一段時間，客戶端程式會取得本地端以及伺服器端之聯絡人列表，並進行同步演算，更新兩端之聯絡人資料庫。

Application Controller 紀錄兩個時間戳記，，分別記錄上次同步後，伺服器端與客戶端的時間。兩邊的時間可能有誤差，故在比較兩邊項目何者較新的時候，會將此誤差計算進去。

首先Application Controller透過ServerContactManager取得伺服器端之聯絡人列表，以及群組列表。再透過WABContactManager取得本地端之聯絡人列表，以及群組列表，如果是本地端新增的聯絡人或群組，CID/GID會以4.3.3.1描述之機制產生。

接著 Application Controller 會依據下列原則比對兩份聯絡人列表：

- 發現某 CID 同時存在於兩邊的列表：
  比對兩邊的 LAST_MODIFICATION_TIME，可能有兩種情況：
  - 伺服器端的 LAST_MODIFICATION_TIME 較新，且伺服器端 LAST_MODIFICATION_TIME 大於 SERVER_LAST_SYNC_TIME：
    表示在上次更新以後，此項目在伺服器端有被修改過，用伺服器端資料更新本地端資料

  - 本地端的 LAST_MODIFICATION_TIME 較新，且本地端 LAST_MODIFICATION_TIME 大於 LOCAL_LAST_SYNC_TIME：
    表示在上次更新以後，此項目在本地端有被修改過，用本地端資料更新伺服器端資料

- 發現一 CID 只存在於伺服器端的列表：
  比對這個項目的 LAST_MODIFICATION_TIME，可能有兩種情況：：
  - 伺服器端 LAST_MODIFICATION_TIME 大於 SERVER_LAST_SYNC_TIME：
    表示在上次更新以後，在伺服器端被新增的項目，將此項目加到本地端

  - 伺服器端 LAST_MODIFICATION_TIME 小於 SERVER_LAST_SYNC_TIME：
    伺服器端此項目在上次更新以後並未被更動，表示是本地端已刪除的項目，從伺服器端刪除該項目

- 發現一 CID 只存在於本地端的列表：
  比對這個項目的 LAST_MODIFICATION_TIME，可能有兩種情況：：
  - 本地端 LAST_MODIFICATION_TIME 大於 LOCAL_LAST_SYNC_TIME：
    表示在上次更新以後，在本地端被新增的項目，將此項目加到伺服器端

  - 本地端 LAST_MODIFICATION_TIME 小於 LOCAL_LAST_SYNC_TIME：
    本地端此項目在上次更新以後並未被更動，表示是伺服器端已刪除的項目，從本地端刪除該項目

更新完聯絡人列表後，Application Controller 依據上述同樣的原則來更新群組列表，但在修改群組資料時，除修改群組名稱外，會一併修改群組內含哪些元素的資訊。然而因為 WAB API 只能知道該群組上次何時被存取，而無法知道某個群組元素何時被加入，故更新群組時，會讓較新一方的群組元素完全取代較舊一方的群組元素。

做完上述之同步動作後，客戶端程式再取得伺服器端以及本地器端之時間，並分別儲存至 SERVER_LAST_SYNC_TIME 以及 LOCAL_LAST_SYNC_TIME 兩個時間戳記。

4.3.5 瀏覽器呼叫伺服器端 API 的方式

在 4.2.4 所描述之伺服器端 API 也可讓瀏覽器直接呼叫，在實作上我們使用 AJAX(Asynchronous Javascript And XML)技術來作為瀏覽器呼叫伺服器端API之方式。

在傳統的互動式網頁裡，瀏覽器透過 POST/GET 傳送變數給伺服器，或從伺服器讀回新的資料時，需要重新載入整個頁面，來獲取由伺服器端產生的，新的頁面。而 AJAX 則是利用 Javascript 類別 XMLHttpRequest，在背景悄悄地與伺服器端通訊，並且可以將傳回的資料用 XML DOM(Document Object Model)解析。再利用 DOM 來動態地修改局部網頁的內容。如此一來不必重新載入整個頁面即可獲取最新的資訊，伺服器端也不用重新產生整個網頁。使用者可獲得較佳的使用體驗，伺服器端的負擔也較小。

考量到以上的優點，本系統選擇在網頁介面中使用 AJAX 介面作為呼叫伺服器端 API 之媒介。伺服器端傳回的 XML 格式回應，可方便地使用 DOM 解析，並動態更新網頁上的資訊。

4.3.6 系統元件互動實例

以下以幾個實例來說明客戶端之各個模組間互動方式。

4.3.6.1 聯絡人狀態同步

當客戶端透過 api_login 登入伺服器端後，會從伺服器端取得聯絡人列表。再利用 MSNChecker 與 SkypeChecker 物件，查詢聯絡人列表中聯絡人即時通訊帳號之上線情形。並利用 api_set_presence 將查詢到的狀況傳送至伺服器端。

每隔一段時間，客戶端會重複上述取得及傳送聯絡人上線狀態之動作，以更新伺服器端上線狀態資料。

4.3.6.2 即時通訊

使用者在 Web 介面上選擇要與哪個聯絡人以何種方式通訊後，控制即時通訊程式的命令會以 URI 的方式傳回給瀏覽器，瀏覽器收到 URI 會根據通訊協定去呼叫對應的程式。

假設現在使用者選擇以 skype 來與 somebody 進行語音交談，則瀏覽器會收到下列的 URI：

skype:somebody?call

因為 skype:是 Skype 客戶端程式向系統註冊的 URI 類型，瀏覽器收到這樣的 URI，會將 URI 轉給 Skype 客戶端程式。而 Skype 客戶端收到後，會打開與 somrbody 的語音交談。

如果是要與 somebody 以 MSN Messenger 進行文字交談，則瀏覽器會收到下列的 URI：

contactto:somebody@msn.com?msntext

在啟動客戶端程式時，客戶端程式會將 contactto:註冊為由我們的客戶端處理。瀏覽器收到這樣的 URI，會 URI 轉給我們的客戶端程式。客戶端程式收到瀏覽器傳來的 URI，從內容判定是要進行 MSN Messenger 文字交談，會使用 MSNController 物件來叫起 MSN Messenger，並打開與 somebody@msn.com 的文字交談視窗。目前支援的 MSN Messenger 交談類型有 msntext（文字交談）、msnvoice（語音交談）、msnvideo（視訊交談）。

第五章 實作成果演示

5.1 客戶端程式

圖 12是我們的客戶端程式，使用者啟動客戶端程式後，可以在「設定」中設定帳號密碼，設定完成後，可以令程式以設定之帳號與密碼登入伺服器。

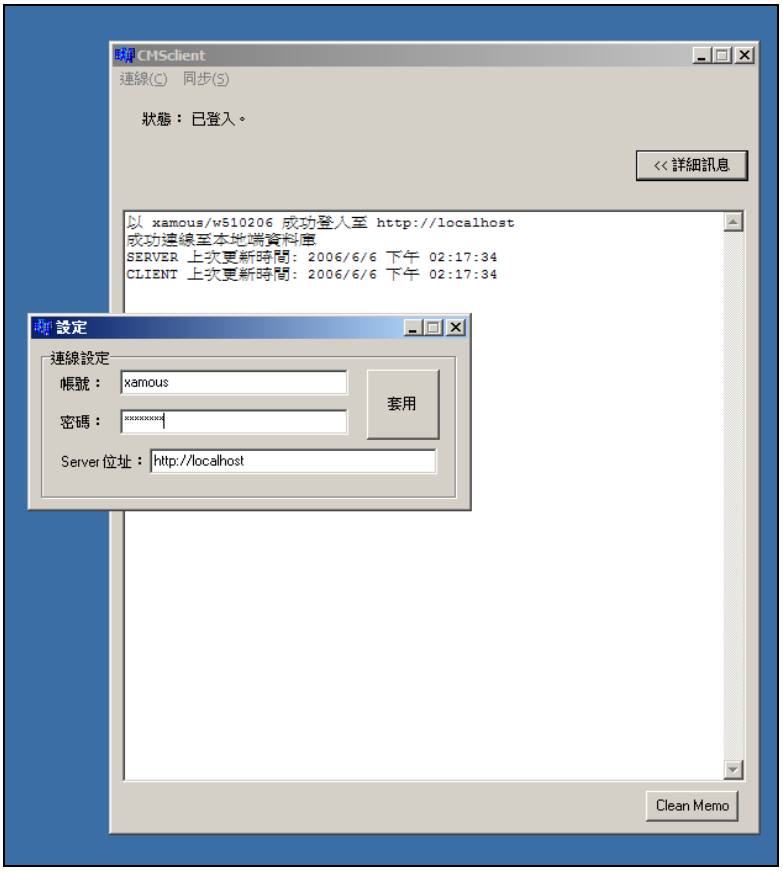


圖12 客戶端程式畫面

登入伺服器後，使用者可選擇「以Server端資料初始化本地端」，以及「以本地端資料初始化Server端」，選擇一邊的資料為基準進行初始化之動作。「開始自動同步」可以讓程式自動同步兩邊聯絡人資料庫之資料，「開始同步聯絡人狀態」則會自動更新伺服器端之聯絡人狀態。「立即同步」則讓程式立即執行同步兩邊聯絡人資料庫之工作（圖13）。

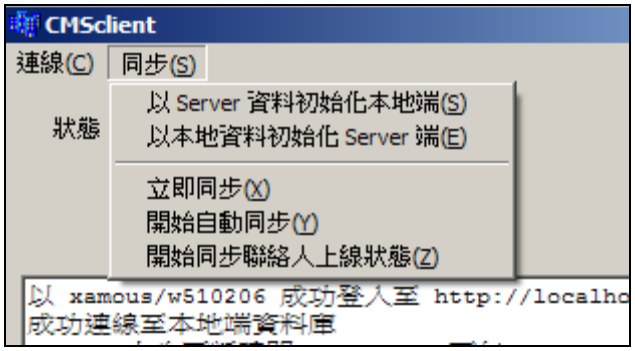


圖13 客戶端程式之功能選單

程式在最小化時可隱藏至系統列，以免佔用桌面空間，使用者仍可直接透過系統列圖示之彈出式選單，來命令程式執行工作（圖 14）。

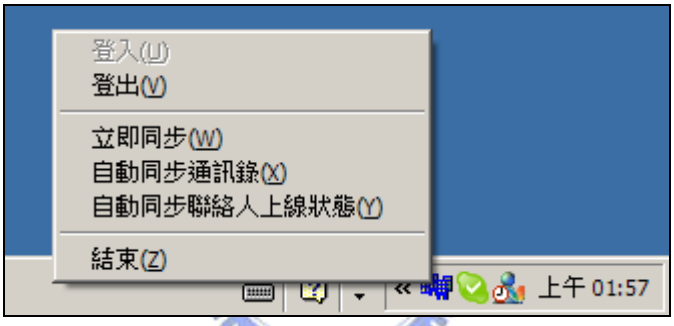


圖14 客戶端程式之系統列彈出式選單

5.2 Web 使用者介面

圖15 Web使用者介面

圖 15為系統提供之Web使用者介面，使用者介面可以分成四個部分：

右上角為使用者資訊區，使用者可以在此設定個人資料、看到有幾個客戶端程式在線上，以及登出系統（圖 15-a）。

第二排的圖示為工具列，從左至右依序為新增聯絡人、新增群組、加入群組元素、移除群組元素、編輯選定物件以及刪除選定項目（圖 15-b）。使用者可直接用滑鼠點選群組或使用者，對於點選的物件執行指定的動作。

左邊為群組資訊區（圖 15-c）。Web介面會顯示所有第一層（不為其他群組所包含）的群組，以及該群組內含之子群組以及聯絡人個數。點選群組會打開該群組，並列出該群組之子群組，並在右邊的聯絡人資訊區顯示屬於該群組之聯絡人，圖 16 顯示使用者開啟「朋友」群組下的「男生朋友」子群組之狀況。再點選一次已開啟之群組，即可關閉該群組。

圖16 已開啟之群組

右邊為聯絡人資訊區（圖 15-d），會顯示目前開啟群組內含之聯絡人。聯絡人資訊以資料卡之形式表現，其欄位說明於圖 17，MSN Messenger與Skype之狀態圖示會隨著該聯絡人之狀態而改變，如圖中之聯絡人，MSN Messenger為「離線」狀態，而Skype則為「離開」狀態。

圖17 聯絡人資料

5.3 存在感知即時通訊

啟動客戶端程式並開始同步聯絡人狀態後，使用者利用瀏覽器登入系統，就可以看到5.2展示之Web使用者介面。使用者可以很方便地存取聯絡人分類以及聯絡人之上線狀態，聯絡人之上線狀態使用與該即時通訊系統客戶端程式相同之圖示來表示，讓使用者可以很直覺地得知聯絡人的上線狀況。

當使用者將游標移動到上線狀態圖示上，Web介面會自動會跳出選單，讓使用者選擇要用何種聯繫方式與該聯絡人聯絡（圖 18）。

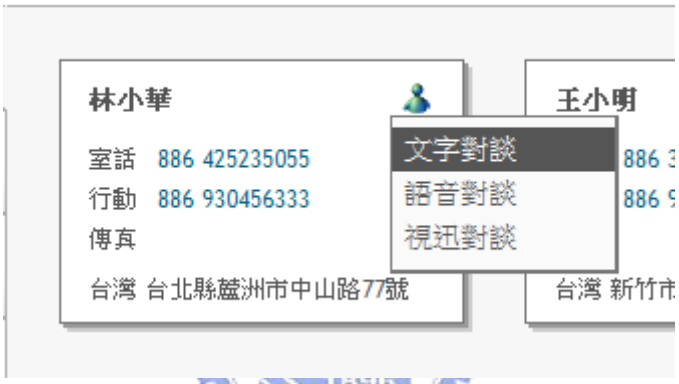


圖18 進行即時通訊之彈出式選單

例如這裡使用者點按了使用MSN Messenger與林小華進行文字交談，則會跳出與林小華交談之MSN Messenger視窗（圖 19）。

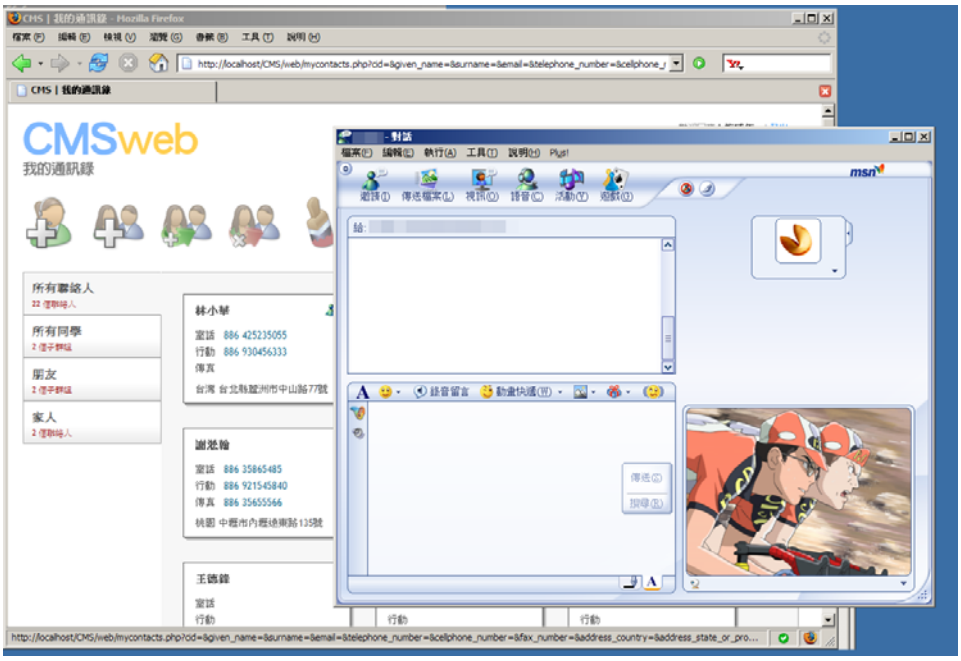


圖19 系統開啟與聯絡人之MSN Messenger文字交談視窗

若使用者選定與對方使用電話聯繫，按下Web介面上該使用者之電話號碼，系統會呼叫Skype，並開始使用SkypeOut撥出該電話號碼（圖 20）。

圖20 系統呼叫Skype使用SkypeOut撥打電話號碼

5.4 聯絡人資料庫同步

使用者啟動客戶端程式之自動同步功能時，系統會自動同步本地端與伺服器端之聯絡人資料庫。

現在使用者在兩部電腦A、B上裝設並啟動我們的客戶端程式，當使用者在電腦A的通訊錄裡增加兩個群組：新群組、新子群組，將新子群組加入新群組為新群組之項目。再新增兩個聯絡人：新聯絡人 1 以及新聯絡人 2。將這兩個聯絡人加入為新子群組之項目（圖 21）。

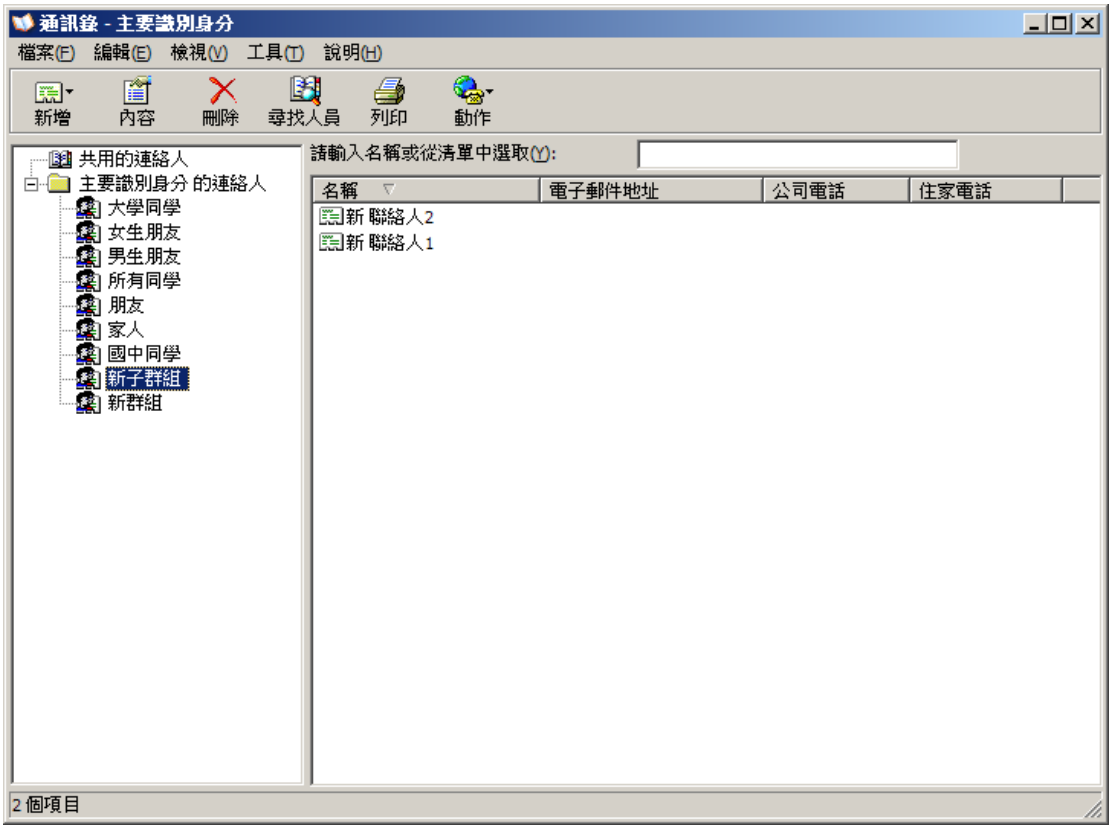


圖21 電腦A本地端通訊錄的變更

當資料庫同步後，使用者在Web介面上已經可以看到剛才對本地端通訊錄進行的變更（圖 22）。

圖22 透過Web介面查詢到同步後的情形

而我們使用遠端桌面連線察看另一部電腦B上之通訊錄，也可以看到剛才對電腦A通訊錄所做的變更，已經更新至電腦B（圖 23）。

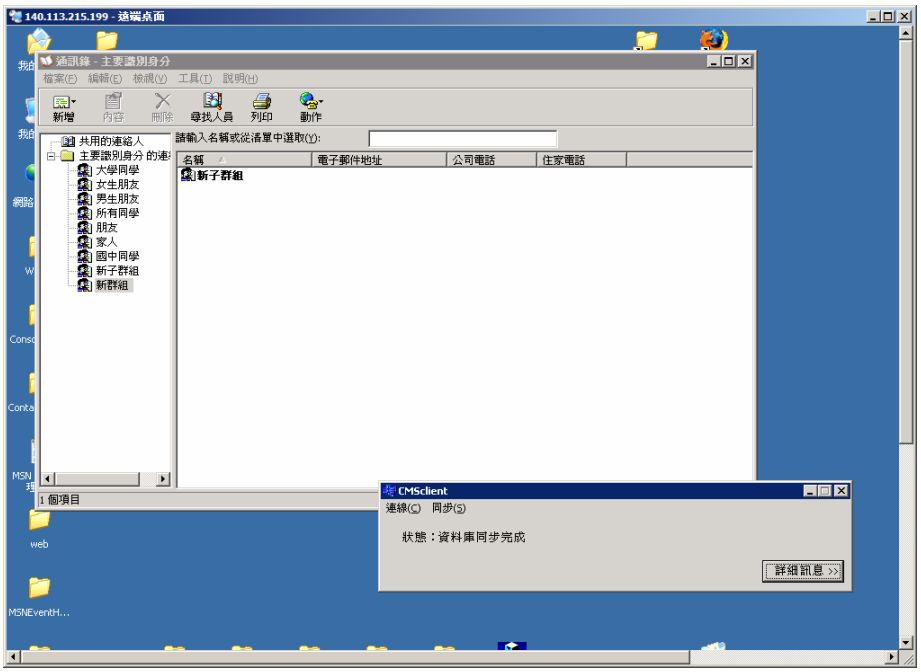


圖23 使用遠端桌面連線察看電腦B之通訊錄

第六章 結論與未來發展

6.1 總結

在這篇論文中，我們構想了一個整合使用者即時通訊上線狀態，並可與系統通訊錄同步之即時通訊與使用者管理系統。並根據我們的需求，設計出一個可擴充的基本架構。在此架構下我們實作出的系統特點有：

- 我們的系統可以讓使用者透過網頁介面，一目了然地查閱自己聯絡人的上線狀態。
- 使用者可以用群組的方式，井然有序地管理自己的聯絡人。
- 使用者可以使用我們的系統同步多台電腦上的聯絡人資料以及群組管理狀態。
- 使用者可以簡單地指定要通訊的對象以及方式，系統可以幫使用者用指定的方法與指定的對象建立聯繫。
- 我們設計出了一個可擴充的架構，可以在將來擴充支援更多種類的的即時通訊系統以及通訊錄系統。
- 我們的系統具備一組通用的 API，不只供系統內呼叫，也可以讓系統與第三方的系統合作，進行其他的服務。

目前的實作，在即時通訊方面我們的系統支援 MSN Messenger 以及 Skype 的上線狀態整合以及呼叫。在通訊錄同步方面，我們的系統實作了與 Windows Address Book 的同步機制。

6.2 系統未來發展方向

6.2.1 即時通訊

在即時通訊方面，目前系統可以支援 MSN Messenger、Skype 的即時通訊，也可以使用 SkypeOut 撥打聯絡人的電話號碼。除了上述兩種以外，還存在有其他各種不同的即時通訊系統，如 Yahoo! Messenger、Google Talk 等等，都可以是本系統將來的擴充支援對象。另外，目前即時通訊需要呼叫客戶端的應用程式，是否可以讓使用者直接透過 Web 介面與聯絡人聯繫，而無須客戶端的即時通訊程式，也是未來發展重點之一。最後，除了網際網路傳訊的通訊方式，也可能支援其他的通訊方式，比如說以手機簡訊與聯絡人聯繫。

6.2.2 存在感知

在聯絡人的存在感知方面，目前我們使用即時通訊上線狀態來做為聯絡人的存在感知依據。在此方面，除了即時通訊以外，使用者的存在狀態是否還有其他的思考方向？比方說對於聯絡人位置的感知（聯絡人目前在某某座標、某某地址），對於聯絡人出現的地點的感知（聯絡人跨進了公司大門、聯絡人進入了會議室），聯絡人在現實世界中活動狀態的感知（聯絡人目前工作中、聯絡人目前睡眠中）…都是可以繼續延伸思考的方向。

6.2.3 聯絡人資料庫

在聯絡人資料庫方面，目前支援的是 Windows 系統上之 Windows Address Book。將來也可以繼續擴充支援其他種類的通訊錄系統，如 Microsoft Outlook、Mozilla ThunderBird…等。同時也可以往其他的作業系統環境發展，比如 PDA 上的 Windows Mobile、或 Linux 的支援等等。

6.2.4 系統應用

在我們的系統裡具備了一組通用的 API 介面，可以供第三方系統來存取。我們希望能採用這個系統來開發出其他的系統，讓我們的系統與其他系統合作，進一步開發出不同種類的服務。

參 考 文 獻

[1] Microsoft MSDN Library/Win32 and COM Development/Messaging and Collaboration, “Windows Address Book”.
http://msdn.microsoft.com/library/en-us/wab/wab/wabentry.asp

[2] Microsoft MSDN Library/Servers and Enterprise Development/Exchange Server, “Messaging API(MAPI) SDK”.
http://msdn.microsoft.com/library/en-us/exchanchor/htms/msexchsvr_mapi.asp

[3] Microsoft MSDN Library/Win32 and COM Development/Messaging and Collaboration, “Windows Messenger”
http://msdn.microsoft.com/library/en-us/WinMessenger/winmessenger/messenger_entry.asp

[4] Skype Developer Zone “Skype4Com Library”
https://developer.skype.com/Docs/Skype4COMLib/

[5] Skype Developer Zone “Skype URI Handler”
https://developer.skype.com/Docs/ApiDoc/Skype_URI_handler

[6] W3C “AJAX Tutorial”
http://www.w3schools.com/ajax/default.asp

[7] Mozilla Developer Center “AJAX: Getting Started”
http://developer.mozilla.org/en/docs/AJAX:Getting_Started

[8] W3C “Javascript Tutorial”
http://www.w3schools.com/js/default.asp

[9] W3C “CSS2 Reference”
http://www.w3schools.com/css/css_reference.asp

[10] 葉秉哲, “物件導向設計模式(Design Patterns)”,培生, 民國九十年

[11] 胡爲君, “DHTML 網頁設計手札(DHTML Utopia: Modern Web Design Using Javascript & DOM)”, 上奇, 民國 94 年

[12] 寧心, “XML 大全”, 華達, 民國九十年