致理技術學院

資 訊 網 路 技 術 系
實 務 專 題 報 告

P2P 檔案傳輸

指導老師：林正平 老師

學生：翟永宏(29534524)

陳庭毅(29534504)

張郁斌(29534502)

何宗憲(29534522)

中華民國 97 年 1 月

致理技術學院

資 訊 網 路 技 術 系

實 務 專 題 報 告

P2P 的檔案傳輸

學生：翟永宏29534524

陳庭毅29534504

何宗憲29534522

張郁斌29534502

本成果報告書經審查及口試合格特此證明。

指導老師:_____

中華民國　97 年　1 月

誌謝

時光飛逝，彷彿昨日才剛踏入致理，這兩年的歲月，心中有許多感謝與不捨，二技生涯中的點點滴滴，不是短短的三言兩語就能表達。除了在課業上有很大的收穫之外，也認識到許多的好朋友。

首先，要感謝我們的指導老師－林正平老師，老師為人務實認真，克盡心思的指導與教誨，使本專題得以豐富充實，在學生們遇到瓶頸時，也適時地提供建議及思考方向，讓學生們可以循著正確的方向邁進。也要特別感謝導師－高楊達老師，在我們製作專題遇到問題時，耐心地指導。此專題製作期間老師們的不辭辛勞，細心地指導我們，給予我們學習與磨練的機會，師恩浩瀚，銘感在心。

此外也要感謝每位組員們的配合與協助，特別感謝組長－翟永宏，在專題製作期間負責監督和指導每位組員的進度，讓組員們能在製作期間內把負責的工作如期完成。由於專題小組們的團結合作，大家互相的努力與勉勵，才能把這艱鉅的任務完成，也在這段期間讓大家明白到團隊合作的重要性。

最後還要感謝家人、同學以及朋友們，在這段期間內所給予的支持與鼓勵。在此，僅對教誨我們、關心我們、鼓勵我們的師長、同學們以及所有全組人員們表達內心最誠摯的感謝。

摘要

在一些應用軟體，例如 NAPSTER 及 Intoned Messenger 受到歡迎後，P2P 的技術開始受到人們的注意。早期的網際網路實際上是一個 P2P 的環境，只是後來隨著網際網路的發展，伺服器為依據的系統才逐漸發展來應付愈來愈多的使用量。但是目前，電腦已經具有愈快的處理速度和較大的儲存量，這使得 P2P 的使用者可透過電腦的直接連結來分享電腦資源和交換資料。

P2P 技術簡單來說，就是藉由系統間的直接交換來進行電腦資訊和服務的分享，最廣為人知的應用模式就是訴訟纏身的 Napster，以及即時傳訊（Instant Messenger）服務，隨著這些服務受到歡迎，P2P 技術也受到多數人的注意。早期的網際網路所面對的是一個集中式系統架構的 P2P 環境，隨著網路蓬勃發展，以伺服器為主的系統必須要能夠承載絡繹不絕的流量。

高速的處理速度與強大的儲存能力是近來電腦所急待解決的發展障礙，單一部電腦必須同時兼具伺服器與使用端的角色，現在使用者利用 P2P 技術，便可透過系統直接連結，彼此互相交換資訊、共享資源。P2P 可以減輕伺服器的負擔，讓系統執行更加順暢，對整個系統而言有相當大的好處。多數的專家也同意，混和式的 P2P/Server 系統對於希

望將 Peer-Based 軟體整合至原有系統與網路的企業，特別有益。

目錄

圖目錄

第一章　緒論

第一節　P2P 傳輸的重要性與發展演進

所謂「P2P」傳輸也就是 Peer to Peer 的簡稱亦是點對點傳輸的意思，P2P 最早的出現在 1999 年 5 月，Shawn Fanning 的 Napster，利用 P2P 的技術以達到音樂檔案的分享。但同年因為版權的問題而遭到官司纏身，此後網路上就出現了許多相似的技術。

其實傳統的檔案傳輸總是必須讓使用者連結至伺服端，透過伺服器的線路來下載檔案，像是這樣的傳輸事實上在伺服端的頻寬負荷相當的重，因為自世界各地不通的下載需求卻都透過相同的幾條線路，就因如此當下載需求大時若伺服端的頻寬不足，很容易就造成傳輸速度的遲緩，且架設伺服器的資金昂貴維護不易所以漸漸的有了所謂的 FTP 檔案傳輸的出現，其實 FTP 並不算是一種 P2P 的方式但是想法很接近，它讓 User 自行下載程式將自己的電腦當成伺服端提供給外界下載，的確這在當有大量載點時候下載需求就能夠被分散，但是分散效果並不平均且架設者在一段時間過後容易將程式關閉，造成載點消失所以日後才漸漸的有了所謂的 BT、驢子、Foxy 等軟體的出現，也就這才正式的進入了 P2P 的時代所謂的 P2P 也就是 Peer－to－Peer 的簡稱，指的是點對點的大量傳輸也就是將只剩少量的資訊透過伺服端來解

決，至於檔案傳輸較佔頻寬的動作是透過使用者直接傳輸給使用者，像這樣的作法就能有效的利用網路中的頻寬且透過續傳的技術可讓一個下載者透過多個載點同時下載同一檔案以提高傳輸速度來解決普遍用戶上傳頻寬較小的問題，亦能有效減少伺服端的負荷伺服端不再需要有大量的上傳負荷當然也就減輕了設備上的需求。

事實上檔案的傳輸在網路中是最佔據頻寬的用途，其實 P2P 的理念就是充分的運用網路上每一條線路，只要還有頻寬就不浪費不再需要讓流量過大的伺服器不斷增添線路，而是有效利用閒置的線路已提高整體網路速率也就是 P2P 檔案傳輸的理念。

(一) 傳統的 P2P 傳輸

P2P 之所以被廣泛運用是因為 P2P 有著點對點傳輸的特性而非以往的主從式網路架構，他有著點與點之間平等的特性也就是使用者不再被區分為伺服端或是使用者，只要對方許可不論是誰都可以藉著這樣的技術傳輸檔案，之所以 P2P 的傳輸模式到日後的今天能成為檔案傳輸的主流格式是因為網路上所流傳的各式軟體如 BT 利用 P2P 的特性設計出操作簡便的傳輸軟體，且巧妙的將相同的檔案切割成數份再由數個不同的使用者同時上傳給同一人後整合，巧妙的克服了非對等式網路普遍上傳速度偏低的缺陷，讓人有檔案傳輸速度加速的錯覺，

雖說這樣的傳輸模式傳輸速度取決於持有相同檔案的人數與他們的上傳速度與分享的意願，相較之下傳統的主從式網路傳輸大部分的伺服器都能有超出一般民眾的上傳速度，但是所分享的檔案卻受到了伺服端的限制若要尋找不同的檔案必須花上大量的心力相較之下由於 P2P 的傳輸建立在多使用者的情形下，想要找到需要的檔案相對容易且由於檔案持續的分散下去持有相同檔案的使用者會越來越多所以下載速度只會隨之加快，相對的傳統的主從式傳輸，傳輸速度會因伺服端上傳速度的不足當出現大量需求者時這些使用者必須平分頻寬而導致下載速度變慢，所以 P2P 的檔案傳輸模式越來越受到推廣相似的軟體也相繼問世逐漸成為檔案傳輸的主流方式。

(二) P2P 特別的應用

在 P2P 發展的初期事實上就有人看到了 P2P 有更廣泛的運用空間，在 1996 年 7 月成立的 Mirabilis 公司並於 11 月份發佈了最初的 ICQ 版本，所謂的 ICQ 也就是英文 I seek you 的諧音這是一套結合傳統主從式網路傳輸與點對點傳輸的通訊軟體，事實上 Mirabilis 所看上 P2P 傳輸的並非檔案傳輸的迅速與便利性而是直接由點對點傳輸訊號不經由伺服器處理更增添了通話內容的保密性，而使用者還是透過傳統的主從式網路與伺服端處理登入或是登出訊息達到管理，事實上這樣的點

子原自於更早的網路聊天室傳統的網路聊天室所有的訊號集中傳輸給伺服端再由伺服端廣播給所有登入的使用者，在早期電腦速度不夠快的時候這樣的作法的確會帶給伺服端的電腦以及頻寬相當大的負擔，且如果伺服端若是有心人士想要竊取聊天內容甚至改寫再傳輸這些存在的問題都是傳統聊天室的隱憂，而早期的 ICQ 所擁有的也只是單純的文字訊號傳輸卻造成了相當大的轟動，紅極一時，相對的此後也有各式相似的軟體紛紛問世如 Yahoo Messenger、MSN 等，且也逐漸的改善當初 ICQ 的不足紛紛的加入了自訂暱稱、狀態顯示、相片分享、視訊等…內容直至發展至今已經成為了 P2P 傳輸的經典類型。

第二節 研究的動機與目的

再專題的開始研究的題目的確困擾著我們，直到老師提供了這樣的題目給了我們做參考我才開始思考這樣的題目究竟有些什麼特色值得我們去研究、探討？在我仔細思考過後便毅然決然的選擇了這個題目，身為資訊網路技術系的我們絕大部分的課程著重於專案開發、網路架構而這個專題剛好將這兩點發揮到淋漓盡致，目前網路的傳輸協定主要分為 TCP(Transmission Control Protocol)與 UDP(User Datagram Protocol)兩種架構,而在第一章所提到的以 P2P 為前提發展的通訊軟體正巧妙的運用這兩種傳輸協定，且既然是通訊軟體則意味著使用者的使用環境不可設限，所以選擇跨平台的程式語言作為專案的開發也是挑戰之一。

P2P 的 P2P 軟體在時下可說是相當盛行，雖說 P2P 的理念是有效的運用網路中的每一條線路，但是事實上隨著 P2P 問世的時間一久也漸漸變的弊病叢生，許多的使用者在下載完軟體後就將程式關閉導致於載點的數量無法有效的擴增，像這樣的問題已經日益嚴重，也漸漸的失去了 P2P 原本的意義雖說在網路上有許多的道德勸導但是始終獲得的回報有限，事實上時下盛行的 MSN 其實也能算是一種 P2P 的軟體，雖說是檔案較小的文字傳輸但是私下聊天的內容不透過伺服器傳

輸，在個人隱私方面又再多添一道防線，由於這樣的特性所以導致每日上線人數高達６００萬人且持續的攀升當中，可惜的是 MSN 的檔案上傳只能一對一的傳輸，普遍民眾的線路上傳並不會太高所以其實透過 MSN 傳輸大量資訊時事實上傳輸速度是屬於較慢。

第三節　研究的範圍

(一)　Eclipse SDK

1.Eclipse 的歷史

Eclipse 是由 IBM 公司所釋出，並公布要給開放源碼社群的一份要價四千多萬美元的禮物。可是 Java 的原創公司卻一點也高興不起來，儘管 Sun Micro 公司不願承認然而在這段時間，Eclipse 卻已經介入了整個 Java 的世界。雖然 Eclipse 是由一個獨非營利組織的 Eclipse.org 來全權管理，不過 IBM 仍扮演著相當的腳色。

2.什麼是 Eclipse?

Eclipse.org 把 Eclipse 看作是一套可以覆蓋所有東西，但不屬於特定類型的一種平台。我們可以用 Eclipse 來開發 Java 程式的做法其實只是 Eclipse 的其中一種而已。

Eclipse 不只是一套開發環境。運用它的 SWT and JFree 程式資料庫，就可當作替代的 AWT 與 Swing 之類的 Java 程式資料庫的選用方案。Eclipse 也供應了一套可以廣泛地運用出各種 Java 應用程式的龐大程式框架。除了 GUI 程式庫以外，我們還可以看到許多種高階的元件，像是 Editor、Viewer、Resource Management、Help System…等等，以及各式各樣的輔助功能。Eclipse 會根據這些元件來創造出 Java IDE 或工作台（Workbench）之類的功能，也可以讓你在自己的應用程式裡頭去取用它們。最特別的是，在 Eclipse 3 所導入的 Rich Client Platform（豐富客戶端平台），將可以為許多類型的應用程式，提供一種共用的程式框架。Eclipse 的『授權型式』允許使用者將這些元件自由的運用到自己的程式當中，並且去修改他們，甚至當成他們自己的部分應用程式進行部署。

(二)Access 資料庫

1. 何謂 Access

Access 是一套 Windows 內的資料庫管理系統。在我們的日常生活上，有著許多事情的管理模式都是已 Access 的方式處理著。

當電腦一步一步演化成圖形介面時，隨之出現許多種類的資料管理系統有 Dbase For Windows、FoxPro For Windows、Access、…等，適用於 Windows 平台之下的資料庫軟體也陸陸續續的推出。其中，具備簡單、方便的操作模式，同時也能完全利用 Windows 之資源者，首推的就是 Access。它是專門用來設計圖形化使用者介面之 32 位元的資料庫管理系統。

2. Access 歷史

Microsoft Office Access（前名 Microsoft Access）是由微軟所發佈的關聯式的資料庫管理系統。它結合了 Microsoft Jet Database Engine 和圖形用戶介面兩項最大特點，是 Microsoft Office 2007 的成員之一。

Access 也是微軟公司的一個通訊程式名稱，創造出想與其他類似相關的程式來競爭。但事後證實這是個失敗的計畫，並且將他終止。數年後他把名字重新命名於數據庫軟體。

這個軟體能夠有效快速的處理大量資料，不過在實際測試中，還是會在某些不同的情況中造成資料遺失。

3. Access 用途

Access 在許多的地方常常被運用，如一些中、小型的企業以及一些較大公司之中的某些部門等等，還有喜愛開發的人員專門用來做出桌面系統。也常常被拿來開發一些簡單的應用程式，而些的應用程式都可利用 ASP 系統在 Internet Information Services 上運行。

因為它的方便和強大的功能為初級的使用者帶來許多的幫助。但是這種過於方便使用的功能常使人誤解。這類的開發者實際上都是沒有受過專業的訓練，因此許多人誤以為這樣的開發人員也可以寫出一套完整的系統，但也有很許多人認為 Access 本身過於方便也產生了這樣的誤導與侷限。

一些較為專業的程式研發員使用 Access 應用開發,主要是給一般的推銷員製作出一些簡單的應用程式工具。但是假如透個網路存取的話，Access 的可擴放性相較之下較為低，因此當許多人同時使用時，他們的選擇大多會傾向於客戶端-伺服器的方案。無論如何，不少 Access 的功能可以運用作其他數據庫的後期應用。

4. Access 特性

Access 的發展，是往簡單不複雜進而容易學習的目標前進，回想當時的 DOS 環境下資料庫軟體，當時這些軟體可是紅遍大街小巷，但是他們卻有一些共同的缺點，在這些系統中不同特性的檔案，每個不同的索引方式就需要一個不同的索引檔，而這些不同的索引檔案卻要由自行設計。如此龐大且複雜的設計，要讓初次使用的使用者立即學會、上手實在有一定的困難。

不過在當 Windows 作業系統出現在市面時，微軟為了因應許多在辦公室環境所需要的功能而發展出了這套 Access（資料庫管理系統），破除了以前使用者需自行規劃資料庫以及編寫程式的規定，使許多初

學的使用者也能快速的上手、使用。

5. Access 模式架構

我們首先透過資料庫的三層模式架構來了解 Access 是如何運作。資料庫的三層模式分別為：外部綱目、概念式綱目及實體綱目。外部綱目和概念式綱目之間提供了概念資料獨立性，概念式綱目和實體綱目之間則是提供了實際儲存資料的獨立性。以下分別簡述：

(1) 外部綱目

可提供不同的應用軟體，以「表單」來建立使用者的操作介面。

(2) 概念式綱目

透過「資料表」來定義儲存資料的屬性，包含欄位的大小，使用的資料型態等，因此包含了資料庫結構的完整資訊

(3) 實體綱目

用來決定資料要儲存於磁碟上或者其他儲存媒體，決定資料的儲存路徑在建立資料庫時就必須決定。

Access 資料庫物件包含了資料表、查詢、表單、報表、資料頁、巨集及模組等物件，這些物件會共用一個資料庫檔案。使用者由表單輸入資料，會儲存於資料表中，再以報表輸出資料；以表單設定查詢條件，從資料表中取得符合條件的資料。如果是在網路上，則以資料頁取得使用者輸入的資料。Access 秀過資料表來儲存資料，並定義資料的相關屬性。每個工作表都會有一個特別的名稱，例如：員工資料表或產品資料表，儲存不同種類的資料。

6. Access 的組成

以 Microsoft Access 組成的共通點來看，其實是有六個不同的物件所組合而成的，這六樣物件分別為資料表(Table)物件、查詢物件、表單物件(Forms)、巨集(Macros)、模組(Modules)、報表(Reports)。

第二章　理論與技術探討

第一節　Peer-to-Peer(P2P)

(一) Peer-to-Peer 的歷史

對於老舊的傳輸方式，透過於 Client/Server 的方式傳輸，代表了你要下載檔案，就一定要有一個 Server 提供下載的服務給你。相對的，這台 Server 要有能力提供給使用者下載，當然，對少數使用者來說，這是很容易的，我們日常生活的個人電腦也能處裡。不過，假如有 100 個使用者要求下載服務時，Server 的工作量就會大量增加，那想想，當有 1000 個、10000 個使用者要求下載服務時呢？那這台 Server 要有足夠的能力，才能處理解決。

從這裡產生 P2P 的概念，Client/Server 的觀念被打破，不要只有一台 Server 提供下載服務，是把每一個 Client 都把它當成 Server。使用者在網路上下載檔案時，會尋找有這個檔案的所有電腦，並求當作 Server 服務使用者。這樣就很快哩，例如說，小明上網尋找想要的檔案下載，程式先是收尋擁有這檔案的所有電腦，結果找到 20 台電腦擁有這個檔案，那你就會收到這 20 台電腦當作 Server 所傳來的檔案，將檔案切割成 20 份，你同時從這 20 台電腦下載這一個檔案。以前從一台電腦下載一個檔案，現在是從 20 台電腦同時下載這份檔案的部份，是

不是就比之前快了 20 倍，更重要的是，這 20 電腦只是假設，或許有這檔案的電腦更有可能是千萬台哩。

(二) Peer-to-Peer 的應用

P2P 技術是從二部電腦為出發，但是若只能二部電腦連結，則其應用的資源相對有限，因此內部網路加上集線器（Hub）、訊號加強器（Repeater）、橋接器（Bridge）等零件，及配合連線拓蹼（Topology）方式就可以使多部電腦同時享有點對點連結技術的功能，一樣可以達到此效果。

目前我們不論在辦公室、學校、家裡，在 Window 系列的作業系統中「網路芳鄰」所使用的內部網路連結，都是以此原理作為出發，發揮點對點連結技術的最大效能。

除此之外，最近在美國十分風行的分享軟體與網站 Napster 和 Gnutella 等，實際就是一個檔案交換平台，做為雙方的橋樑，使下載的一方連到提供檔案的伺服器上，也是源自於 Peer to Peer 的概念，被認為是 B2B、B2C 之外的另一種電子商務模式，一般人稱之為 P2P（Peer to Peer），網友們使用分享軟體時，也不要忘記這個同名的連接技術！

(三) Peer-to-Peer 的特色

P2P 點對點連結技術必須具備至少二部電腦及 PCI 界面網路卡、RJ-45 雙絞線接頭等相關的配備，在 Window 作業系統即可安裝。P2P 點對點連結技術主要特點在於資源分享以及經濟實惠。

將二部電腦以簡單的網路連線連結，是內部網路的基本原理。它最實用的地方就是「資源分享」，所謂資源分享是指 P2P 可以使檔案、硬碟共享，並且使用共同的電腦周邊設備如可以共享一台印表機、一台掃描器、一台 CD-ROM 等。

在省錢方面，P2P 點對點連結技術是一種非常簡單的連結方式，大約只要花費三、四千元就可以使二部電腦相互連結。而且主要是利用 RJ-45 雙絞之四組（4 Pairs）訊號線進行同時接收、發送的全雙工（Full Duplex） 特性，如果你只有一台印表機、或是二部印表機的功能不同（如黑白與彩色），透過 P2P 的內部網路連線方式，可以使二部電腦同時共用一台印表機或是二部電腦都同時可以使用功能不同的印表機，達到資源共享，不需要擴充 PC 周邊設備的限制，因此可以節省電腦周邊的使用成本。

(四) Peer-to-Peer 的優缺點

P2P 的優點，剛剛上面也提到了，除了可以減少 Server 的負擔，

也增加了檔案分享的快速性。而缺點，就是隨之而來的網路安全。你想想同時對上 100 台、1000 台電腦連線，使用者的電腦就相對開了許多的 Port，這對駭客要侵入你的電腦，簡直是輕而易舉，加上網路攻擊的一種--洪水攻擊,只要對你的 Port 大方的提出要求說要傳送檔案送進你的電腦，這個 Port 如果開了超過你的電腦可以負荷的程度，你的電腦也就忙不過來了。另一個缺點,就是智慧財產權問題。以往用 Server 下載非法軟體，這樣說吧，就是比較慢；但是現在，利用 P2P 你可以盡情的分享你的檔案給你的同好，這樣更是讓這種情形越演越烈。

(五) Peer-to-Peer 的安全性

這是一體兩面的問題，有關網路的安全性上面那一段也提過，使用 P2P 的技術，相對的你的電腦也是 Server 的一份子，這對網路攻擊者可是有機可乘。使用者提供的 Port，無論是對內對外都比以往相對的多，很容就被植入木馬或是惡意程式，這也是學術網路、學校機關的網路都會禁止使用 P2P 一類的軟體，在這方面的網路安全還未發展完全時，這是唯一的辦法。

第二節　Java 程式語言

(一)　何謂 Java

我們所使用的 Java 是一個能在不同平台上工作的程式語言，為了讓 Java 的程式碼可以在你的平台上執行，必須安裝所謂的 Java Virtual Machine，其功能就是將 Java 虛擬程式碼轉換成可以實際在你平台上執行的程式碼。

Java 特點有物件導向的功能、完全支援網際網路、擁有豐富的函數庫、更重要的是擁有跨平台的功能。

它是先經過編譯出來在直譯的機器碼，稱為位元碼(Byte-Codes)。透過 Java 的直譯器(Interpreter)便可解譯並執行。而 Byte-Codes 最大的好處就是可跨平台！就因為他的跨平台的特性！使它應用於各個作業平台，也使它急速的普及！

(二)　Java 的好處

1.Surmounts The Platform：

簡單來說 Java 不只能在 Windows 上工作,還能在比較知名的平台作業，例如 Linux 等，除了一般電腦平台外，現在也有很多電子產品也用 Java 操作，例如我們現在在用的手機、全民健保卡、信用卡、金融卡等，Java 能在這些東西上使用，最主要是 Java 有兩個要素，第

一是 Java 虛擬機器、第二是 Java 應用程式介面 Java API、JVM 簡單的說就是 Java 的編譯器，把 Java 編譯成 Class 碼，然後在轉換成機械碼，所以 Java 所寫的程式，只要電子產品有 JVM 的系統，就可以使用 Java，至於 Java API 就是有大量的軟體元件,他提共了很多功能,例如 GUI 主件 ,而 Java API 內還將相關的類別和介面做分類,形成一個個的資料庫,Java API 和虛擬機器將程式和硬體分開。

2. Simple：

Java 比 C++容易多了，例如：多重繼承和指標，這些在 Java 裡面已經都沒有了，Java 移除了指標的問題，用介面代替多重繼承，而記憶體管理也改為自動管理，不用人力來操作。

3.Thing guidance：

Java 是一個標準的物件導向程式，和 C++一樣有類別屬性物件方法封裝等。

4. Security：

當 Java 程式開始工作時,JVM 就會開始監視他,只要有一有異常，JVM 就會馬上制止他，可以簽署數位簽名，也可以自行調整安全度，

有這樣的設計，方能在現在的網路世界裡確保安全性。

5. Stable：

有些程式會在執行時發生錯誤，使得程式突然地消失，在前面說 Java 把去除指標型別，因為指標行為他容易發生錯誤行為，而且 Java 他有特別的錯誤防制系統，可以攔截使用者操作的錯誤，比如說輸入資料行號不符等以確保程式穩健執行。

6. Multi-Executions：

多執行緒的意思是，一個程式可以同時執行多項工作能力，例如可以同時把網路下載影片也要同時放映，對 Java 而言多執行緒是自動控制的,而對其他程式,確需勞頓系統指派，多執行緒對圖形介面和網路應用程式特別好用。

7. File Operation：

Java 提供了很多的檔案輸出入的系統，不管什麼型別，都有相對應的 I/O。這些 I/0，都是以類別模式存放，只要程式設計師有需要，就可以到 I/O 類別程式庫去尋找。

8. AWT And Swing：

這兩個是 Java 圖形介面系統，Java 可以以文字方法輸出，當然也可以以圖形介面輸出，可是 Java 不像微軟的 VB New VC#等系統，必須要由程式設計者自行設計，AWT 是 Java 早期的圖形介面，從 Java 1.0 時就有的程式介面，可是他的效果不是很好，後來 Sun 幫 Java 設計以 Java 系統的 Swing，Swing 的表現比 AWT 好，而且變化比較多，是目前視窗常用的系統。

(三) Java 如何解決速度過慢

Java 硬體加速方案約在 5 年前首次出現，這些方案嘗試在主要處理器旁加裝一個小型處理器來專責執行 Java 程式，希望藉此解決效能問題。值得一提的是運用了硬體加速器後，Java 的執行速度確實提升了，而且它不需任何額外的記憶體－在 5 年前針對手機來說，這是一個很重要的優點。其缺點則是須使用額外硬體及功率消耗的增加。

在現今技術的發展下，即使針對手機來說，記憶體容量需求的問題已經不復存在；以往受到記憶體數量限制的系統（主要是手機）已不再有這類問題。硬體加速器的其中一項優勢也就因此消失。另外一個對硬體加速器不利的就是軟體加速技術經歷長久的研發，已經得到了相當的進展。在效能方面已遠遠超越硬體加速器，且沒有硬體解決

方案須使用額外硬體與增加功率消耗的問題。

為了分析軟體解決方案的效能如何超越舊型硬體方案，我們必須先回顧 Java 的演進歷史。最早的 Java 建置方案是由一套轉譯程式，將每個 Java 指令都轉譯成對等的微處理器指令，並根據轉譯後的指令先後次序依序執行，由於一個 Java 指令可能被轉譯成十幾或數十幾個對等的微處理器指令，這種模式執行的速度相當緩慢。

針對這個問題，業界首先開發出 JIT（Just In Time）編譯器。當 Java 執行 Runtime 環境時，每遇到一個新的類別(Class：類別是 Java 程式中的功能群組)，類別是 Java 程式中的功能群組－JIT 編譯器在此時就會針對這個類別進行編譯（Compile）作業。經過編譯後的程式，被優化成相當精簡的原生型指令碼（Native Code），這種程式的執行速度相當快。花費少許的編譯時間來節省稍後相當長的執行時間，JIT 這種設計的確增加不少效率，但是它並未達到最頂尖的效能，因為某些極少執行到的 Java 指令在編譯時所額外花費的時間可能比轉譯器在執行時的時間還長，針對這些指令而言，整體花費的時間並沒有減少。

基於對 JIT 的經驗，業界發展出動態編譯器，動態編譯器僅針對較常被執行的程式碼進行編譯，其餘部份仍使用轉譯程式來執行。也就是說，動態編譯器會研判是否要編譯每個類別。動態編譯器擁有兩項

利器：一是轉譯器，另一則是 JIT，它透過智慧機制針對每個類別進行分析，然後決定使用這兩種利器的哪一種來達到最佳化的效果。動態編譯器針對程式的特性或者是讓程式執行幾個循環，再根據結果決定是否編譯這段程式碼。這個決定不見得絕對正確，但從統計數字來看，這個判斷的機制正確的機會相當高。事實上，動態編譯器會根據「歷史資料」做決策，所以程式執行的時間愈長，判斷正確的機率就愈高。以整個結果來看，動態編譯器產生的程式碼執行的速度超越以前的 JIT 技術，平均速度可提高至 50%。

但軟體加速如何超越硬體加速，尤其是特別設計的「硬體加速器」？我們可以從下文了解原因。

開發 Java 加速器時，工程師可專注於開發硬體的小巧，也可專注於加快處理速度。為了 Java 的可攜性，讓 Java 程式碼能在所有硬體架構上執行，Sun 設計的 Java 指令結構有異於其他硬體架構，因此要開發一套 Java 加速器是相當複雜的工作。工程師不可能將硬體加速器設計得過大，因為這會使該加速器失去競爭力，工程師只好犧牲加速的功能來縮減加速器的尺寸。這類加速器在執行 Java 程式時的速度超過一般簡易的轉譯器，但比不上如動態編譯器的先進的軟體加速器，以目前先進軟體加速器的普及，拿簡易的軟體轉譯器來做比較已經沒有

意義。

軟體方案速度較高的另一項原因是因為處理器是在原生模式下執行程式碼，因此若您針對某種處理器編譯 Java 程式碼，處理器就能在最高速度的模式下執行該程式。動態編譯器能提供接近原生模式的效能，並僅針對重要的程式區段來作編譯。硬體加速器並未以處理器原生模式來執行程式碼，而仍是以 Java 程式來執行。

Java 指令架構在效能提升方面造成許多障礙：首先，它不允許暫存器，因此所有的運算都是在堆疊（Stack）中操作。運算用的資料及結果只能放在堆疊中，運算時再從堆疊中取出使用。Java 指令架構亦不允許 C 語言所支援的指標（Pointer）。若在一個類別中有某個數值，我們希望讓另一個類別使用該數值時，在 C 語言中僅須用一個指標指向記憶體中儲存該數值的位址。因為指標若誤用可能造成記憶體資料錯亂，Java 指令架構為維持程序執行的穩定度，因此不允許使用指標，其代價就是必須使用更多的步驟來移動資料。在傳遞數值時，就算該數值不會有改變，也必須從一個記憶體位址複製到另一個記憶體位址。這種方式的執行速度遠低於只傳遞一個會指向資料所在地的指標。

上面所提的這些障礙都是硬體加速器無法解決的。因此，儘管硬體加速器出現已有 4、5 年的時間，卻不像軟體解決方案一樣已經有數

個世代的改進，現在其效能已經遠落後軟體加速器。

以往手機所面臨記憶體容量有限的困境，因為 Java 編譯器會耗盡所有記憶體，因此手機無法搭載任何編譯器資源，在這樣的環境下，當硬體加速器在當年（1999）問市時，對手機運用而言，它合理避免了對記憶體容量額外負擔的問題。

最後一點，現今的手機幾乎都搭配彩色螢幕，支援富含大量繪圖元素的應用。手機微控制器遇到最大的問題是繪圖加速，而不是 Java。若廠商所設計的產品僅搭載簡單的螢幕以及少量的記憶體，那就可能需要一個 Java 硬體加速器，但在大多數狀況下，軟體解決方案都遠遠超過硬體解決方案。

第三節　TCP/IP 系統

「TCP/IP」是 Transmission Control Protocol (TCP) 和 Internet Protocol (IP) 的簡稱，為網路上的一種通訊協定，是上網時大家都遵循的一些規則。有了這些規則，即使是不同的電腦設備與作業環境，都可以透過這些通訊協定來互通訊息。同時也是因為這些規則，Internet 才可能有這麼多應用 (WWW、E-Mail、FTP、Telnet 等)。TCP/IP 是一個開放的標準，任何人均可自由的下載和 TCP/IP 相關的技術標準和文件；同時 TCP/IP 也鼓勵使用者評論這些標準，一同來幫忙改善

網路的執行效率。

TCP/IP 通訊協定所使用的文件、技術以及通訊協定標準（任何 Internet 上的技術），都會經過實驗性、推薦使用、評論修訂、標準化，四個步驟來形成，而這些步驟皆是經由 Internet RFC (Requests For Comments) 來實行。RFC 提供了一個實驗性的環境，將新的技術、文件、通訊協定推薦出來，再經過各種專家的評論調節之後，發表出標準化的技術、文件及通訊協定。

第四節　UDP 系統

用戶資料訊息協定 (User Datagram Protocol, UDP)是一個簡單的傳輸層協議，IETF RFC 768 是 UDP 的正式規範。

在 TCP/IP 模型中，UDP 為網路層以下和應用層以上提供了一個簡單的介面。UDP 只提供資料的不可靠傳遞，它一旦把應用程序發給網路層的資料發送出去，就不保留資料備份（所以 UDP 有時候也被認為是不可靠的資料訊息協定）。

UDP 開頭欄位由 4 個部分組成，其中兩個是可選的。各 16Bit 的來源埠和目的埠用來標記發送和接受的應用進程。因為 UDP 不需要應答，所以來源埠是可選的，如果來源埠不用，那麼置為零。在目的埠後面是長度固定的以位元組為單位的長度域，用來指定 UDP 資料訊息包括資料部分的長度，長度最小值為 8Byte。

由於缺乏可靠性，UDP 應用一般必須允許一定量的封包、出錯和複製。有些應用，比如 TFTP，如果需要則必須在應用層增加根本的可

靠機制。但是絕大多數 UDP 應用都不需要可靠機制，甚至可能因為引入可靠機制而降低性能。流媒體、實時多媒體遊戲和 IP 電話 (VOIP) 就是典型的 UDP 應用。如果某個應用需要很高的可靠性，那麼可以用傳輸控制協議來代替 UDP。

第五節　SHA-1 系統

最初載明的演算法於 1993 年發佈，稱做安全雜湊標準 (Secure Hash Standard)，FIPS PUB 180。這個版本現在常被稱為 SHA-0。它在發佈之後很快就被 NSA 撤回，並且由 1995 年發佈的修訂版本 FIPS PUB 180-1 (通常稱為 SHA-1) 取代。SHA-1 和 SHA-0 的演算法只在壓縮函數的訊息轉換部份差了一個位元的循環位移。根據 NSA 的說法，它修正了一個在原始演算法中會降低密碼安全性的錯誤。然而 NSA 並沒有提供任何進一步的解釋或證明該錯誤已被修正。而後 SHA-0 和 SHA-1 的弱點相繼被攻破，SHA-1 似乎是顯得比 SHA-0 有抵抗性，這多少證實了 NSA 當初修正演算法以增進安全性的聲明。

SHA-0 和 SHA-1 可將一個最大 264 位元的訊息，轉換成一串 160 位元的訊息摘要；其設計原理相似於 MIT 教授 Ronald L. Rivest 所設計的密碼學雜湊演算法 MD4 和 MD5。

第三章　本專題系統

第一節　系統架構

如圖 3-1 就是註冊帳號，還有與 Server 端確認帳號是否被使用。

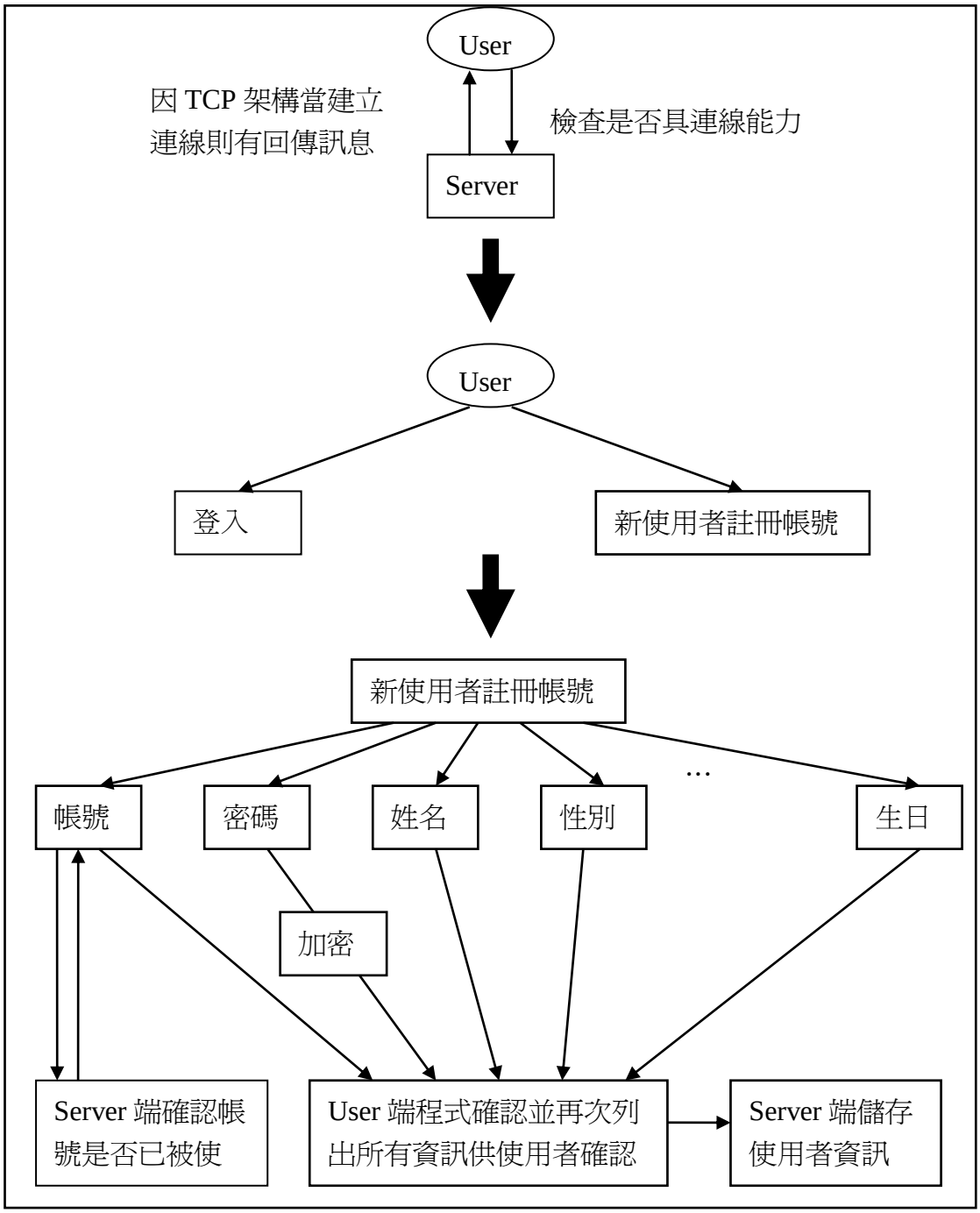

圖 3-1 系統架構圖(新使用者註冊)

如圖 3-2 這裡是帳號密碼登入的加密過程。

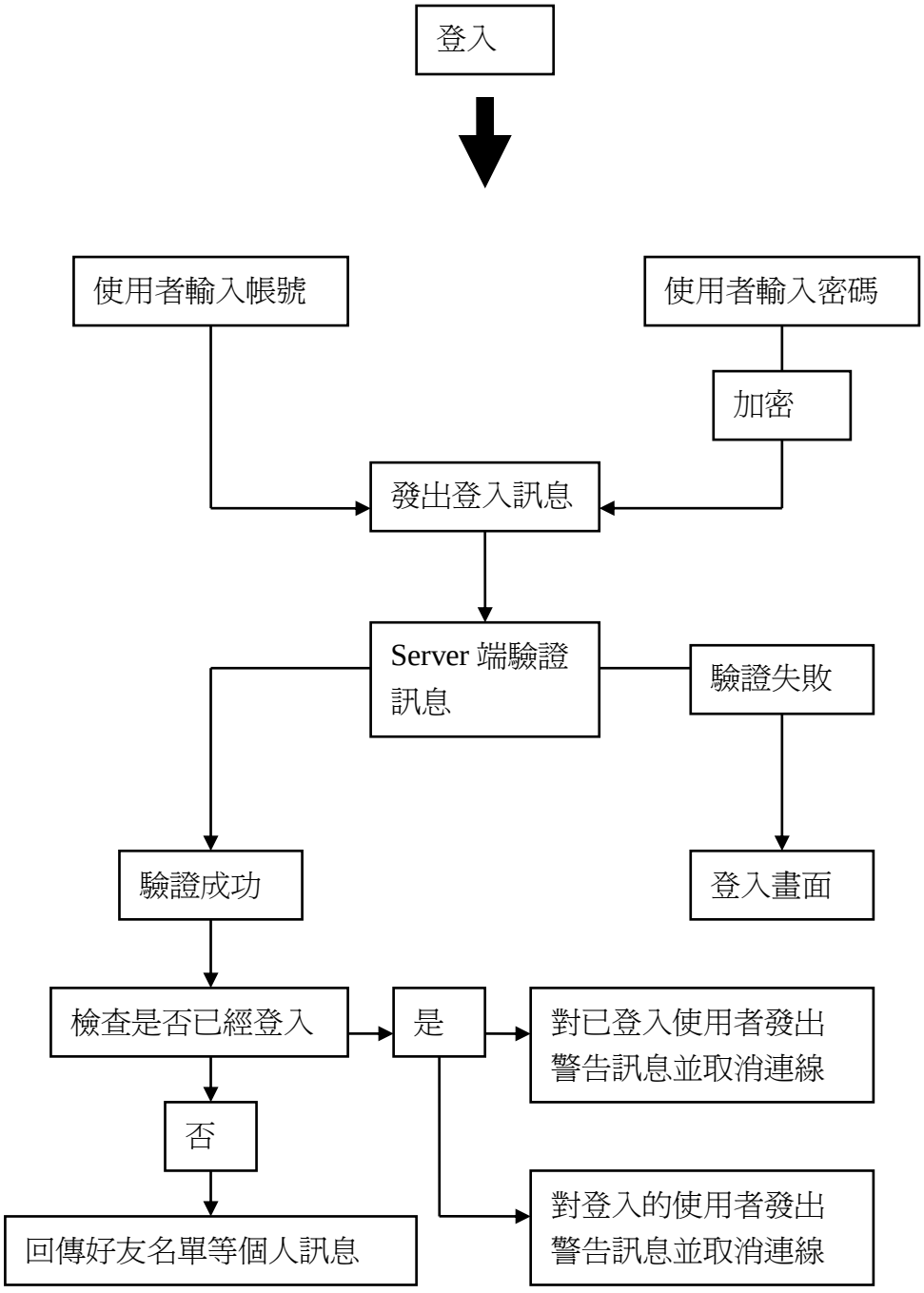

圖 3-2 系統架構圖(登入)

第二節　系統功能簡介

(一)　新使用者註冊

提供第一次使用此系統者建立一個新帳號，讓使用者可以用此帳號登入此系統。

(二)　登入

在此輸入註冊時所建立的帳號與密碼,即可登入此系統開始使用。

(三)　登出

離開現在所使用的帳號，方便不同帳號登入或離開此系統。

(四)　好友登入提示

當您登入時，你的聯絡人清單中若有人上線則系統將會發出訊息提醒你，讓您清楚知道好友上線與否。

(五)　自訂暱稱

讓你在登入帳號時可隨時改變的名稱，當您尚未輸入此暱稱時自動顯示您的帳號讓其他線上聯絡人觀看，若輸入暱稱時則線上聯絡人將看到的是您的暱稱。

(六)　自定狀態

提供（線上）、（忙碌）、（離開）、（通話中）、（外出用餐）、（馬上回來）及（顯示為離線）這些狀態，讓其他連絡人觀看而淺顯易懂。

(七) 自訂相片

提供使用者可擺放自己喜歡的照片或圖片供線上朋友觀看，也方便讓線上其他人清楚的知道 Who are you。

(八) 新增好友

可在此處加入聯絡人帳號，方便與好友交談與討論，而有新增好友的聯絡人系統才可提供（好友登入提示）此功能。

(九) 刪除好友

將已加入的好友與聯絡人從清單之中刪除。

(十) 傳送訊息

可在此視窗方塊中輸入你所要讓對方知道的文字訊息，借此傳送至對方談話視窗中讓對方清楚明白你所要表示的東西。

(十一) 改變字型

可更換你所想要的字型（如標楷體、新細明體...等等）以及改變文字的大小、顏色、粗體、斜體...等

(十二) 震動

搖動此視窗再搭配提示音效提醒對方，讓對方感受到你有要事須找他交談、討論。

(十三) 檔案傳輸

可在此傳送你所要給對方的所有資料以及檔案，讓雙方可運用此系統快速的傳遞資料或檔案給對方。

(十四) 伺服端管控

處理當使用者重複登入時，對正在線上的使用者提出離線提醒，然後發出訊息警告使用者密碼有外洩的可能。資料的保密，保護所有使用者防止資料、對話外洩被竊取的可能性。

第三節　系統特色與技術

(一) UDP

1. 何謂 UDP

所謂的 UDP(User Datagram Protocol)就是簡單的傳輸層協議，IETF RFC 768 是 UDP 的正式規範。

在 TCP/IP 模型中，UDP 是在網路層以上應用層以下。UDP 提供的資料是不可靠傳遞，一旦把應用程序發給網路層的資料傳送出去，就不保留資料備份（所以 UDP 有時也被認為是不可靠的傳輸）。

2. UDP 架構

UDP 開始欄位是由 4 個部分所組成，其中可選的是兩個。各 16Bit

的來源埠和目的埠用來標記發送和接受的應用。因為 UDP 不需要回應，所以來源埠是可選擇的，如果來源埠不用，那麼歸為零。在目的埠後面是固定長度以位元組為單位，用來指定 UDP 封包包括資料部分的長度，長度最小值為 8Byte。

但因又缺乏可靠性，UDP 應用一般必須允許一定量的封包複製和錯誤。有些應用，比方 TFTP 如果他需要則必須是在應用層增加根本的可靠機制。但是絕大多數 UDP 應用都不需要可靠機制，甚至可能因為引入可靠機制而降低性能。多媒體遊戲和 IP 電話就是標準的 UDP 應用。假如某個應用需要很高的可靠性，那麼可以用傳輸控制協議來代替 UDP。

3. UDP 的缺點

因為缺乏壅塞控制（Congestion Control），需要網路的機制來減小因失控或高速 UDP 流量負荷而導致的壅塞崩潰效應。換句話說，因為 UDP 發送者不能夠檢測壅塞，所以像使用拋棄技術的路由器這樣的網路設備常常就成為降低 UDP 過大通信量的有效工具。封包壅塞控制協議設計成通過在如媒體類型的高速率 UDP 流中增加主機壅塞控制來減小這個潛在的問題。

典型網路上的眾多使用 UDP 協議的關鍵應用一定程度上是相似

的。這些應用包括 DNS、簡單網路管理協議動態主機配置協議和路由信息協議等等。

(二) TCP

1. TCP/IP 的起源歷史

以往的電腦並不是我們日常生活所看見的那樣迷你的 PC；他們大部分是一個中央控制系統，用線路與終端系統(輸入輸出設備)連接起來。這樣的一個連接系統，就是網路的最原本樣式。各自的網路都使用自己的一套規則協定，可以說是互相沒什麼關係。

用來在美蘇冷戰期間，美國政府機構為了能夠連接各個不同地方的網路系統，以應付各式各樣的危急狀況。這個計劃，是由 Advanced Research Project Agency 發展的 Arpanet 網路系統，所要研究的就是當某部份電腦的網路遭到不明攻擊而癱瘓時，可嘗試透過其他不同的網路線路來傳送資料。

Arpanet 這個理念和想法，不但成功的研發出一套相當可靠的資料通訊技術外，同時還兼顧了可跨平台作業使其大大增加它的方便度。後來，Arpanet 的計畫相當成功，進而造就了今日所看到的網際網路。

2. TCP/IP 的應用

TCP/IP 可用在各個網路通訊，包括了家庭、校園、公司以及全球 61 個國家實驗室，這些大機構都投注了許多的資源用來開發和應用 TCP/IP 網路。

這個技術的應用，讓所有的研究人員能夠透過網際網路和全世界相同研究領域的人們一起分享資料或探討問題。網路如今已經印證了 TCP/IP 的可行性與它優秀的整合性，它能適應現在各種不同的網路技術。對現在而言，現今網路發展的局面，TCP/IP 可說是一種不可或缺的成就。

TCP/IP 不但成功的連接起不同區域的網路，且後來許多開發的應用程式和觀念也幾乎的是以 TCP/IP 為基礎觀念而研發而生的，進而給許多不同的廠商能夠不必侷限硬體的結構而開發出共通的應用程式，像現今廣泛應用的 WWW、E-Mail、FTP、DNS 服務等等。

3. TCP/IP 特性

Connectionless Packet Delivery Service：

是一種封包交換的模式。TCP/IP 依系統信息中的位址資料來進行傳輸，但它卻不能確保每個獨立的封包的可靠性以及送達目的地的順序之正確。在所有的連線過程中，線路都不會被獨自佔據，反而是以直接對映在硬體位址上，所以比較會有成效。最大的特點就是這種封

包的傳送交換方式，讓 TCP/IP 可以適應所有不同的網路硬體。

Reliable Stream Transport Service：

封包交換的可靠性並不高，因此我們需要利用某些軟體來進行偵測和修復傳送封包中可能出現的錯誤，和處理一些較不完整的不良封包。這樣的功能是要用來確保電腦與電腦之間能夠建立、連接以及傳送較大量的資料。最重要的技術也就是將所謂資料流一段一段進行分割，再進行編號進而傳送出去，然後再經由接收方進行確認來保護資料的安全，提高資料傳送中的可靠性。

Network Technology Independent：

在封包交換技術中，TCP/IP 是一套獨立的系統。TCP/IP 有自己的一套資料包規則和定義，能應用在不同的網路之上。

Universal Interconnection：

只要電腦用 TCP/IP 連接網路，都將獲得獨一無二的識別位址。資料包在交換的過程中，是以位址資料為依據的，不管封包所經過的路由之選擇如何，資料都能被送達指定的位址。

End-to-End Acknowledgements：

TCP/IP 的確認模式是以“端到端”進行的。這樣就無需理會封包交換過程中所參與的其它設備，發送端和接收端能相互確認才是我們關心得。

Application Protocol Standards：

TCP/IP 除了提供基礎的傳送服務，它還提供許多一般應用標準，讓程式設計人員更有標準可依，而且也節省了許多不必要的重複開發。

正式由於 TCP/IP 具備了以上那些有利特性，才使得它在眾多的網路連接協定中脫穎而出，成為大家喜愛和願意遵守的標準。

4. TCP/IP 在網路中所扮演的角色

TCP/IP 的全名叫做 Transmission Control Protocol / Internet Protocol (TCP/IP)，原本是用來配合 Arpanet 來處理不同硬體之間的連接問題的，比如 Sun 系統和 Mainframe、Mainframe 和個人電腦之間的連接。

Internet Protocol (IP) 工作於網路層，它提供了一套標準讓不同的網路有規則可循，當然，前提是您想使用 IP 從一個網路將封包路由到另一個網路。IP 在設計上是用來在 LAN 和 LAN 及 PC 和 PC 之間進行傳輸，每一台 PC 或每一個 LAN，都可以由一組 IP 位址來區分。一個

IP 位址的格式是四個用小數點(.)分隔開來的十進位數字，每各數值介乎於 0 到 255 之間。實實上，每一組數字，在 IP 位址中是以“Octet”的格式承現的，也就是完整的 8 個 bit。我們會在後面的「網際網路層」中詳細講解 IP 位址的所包含的信息和功用。

您可以把 IP 看成是遊戲規則，而 TCP 則用來詮釋這些規則的，更準確來說，TCP 在 IP 的基礎之上，解釋了參與通訊的雙方是如何透過 IP 進行資料傳送的。TCP 提供了一套協定，能夠將電腦之間使用的資料透過網路相互傳送，同時也提供一套機制來確保資料傳送的準確性和連續性。

雖然 TCP/IP 原先是專門為幾所大學和機構的使用而設計的，但現在 TCP/IP 已經成為最流行的通訊協定了，我們使用的 Internet 就是用 TCP/IP 來傳送封包的。

(三) SHA-1

以下是我們系統加減密的流程簡介

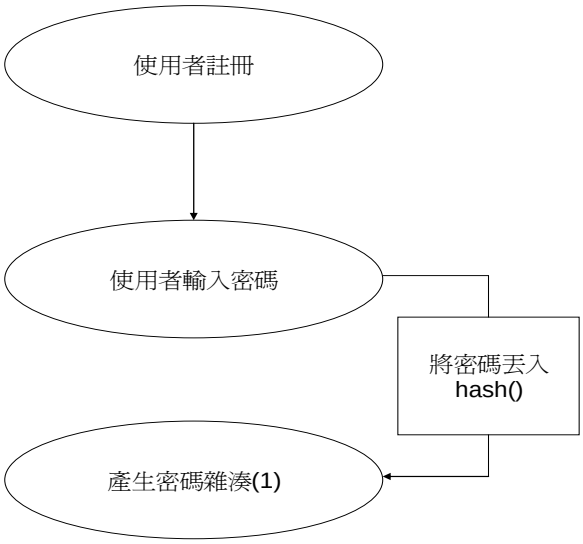

圖 3-3 SHA-1 圖形流程簡介（1）

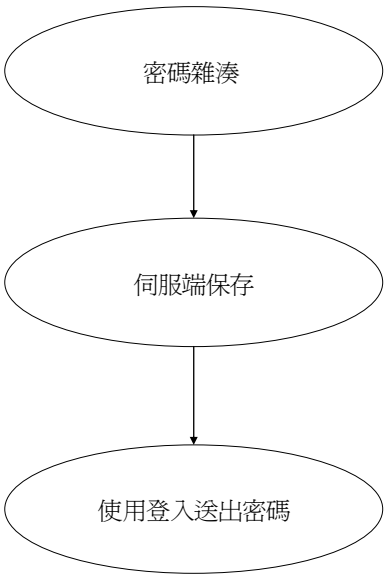

圖 3-4 SHA-1 圖形流程簡介（2）

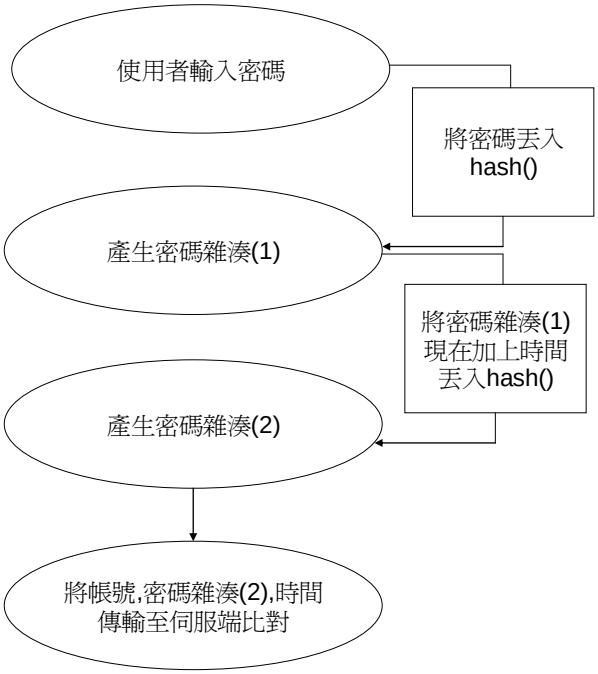

圖 3-5 SHA-1 圖形流程簡介（3）

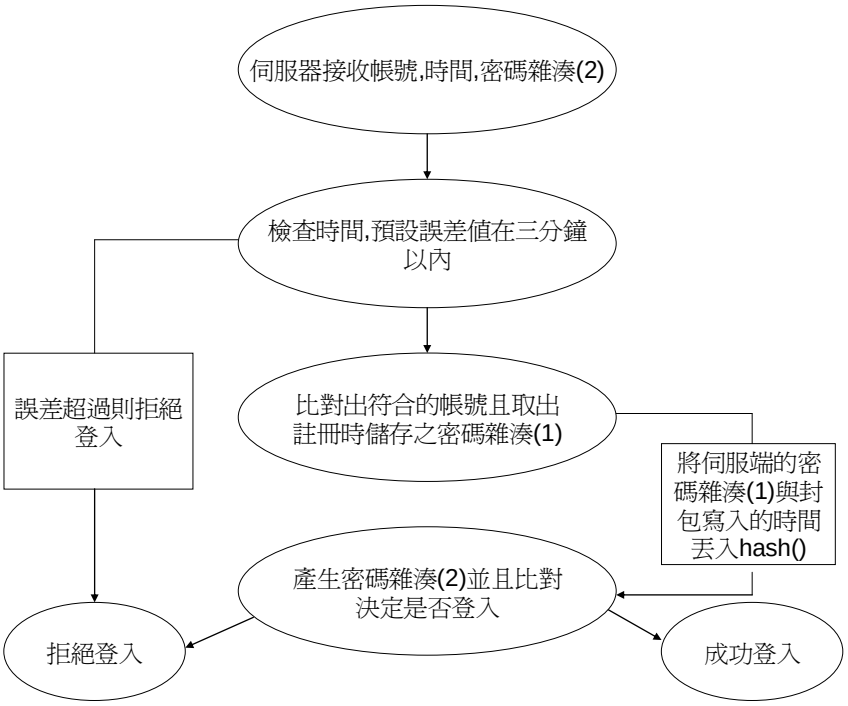

圖 3-6 SHA-1 圖形流程簡介（4）

意外的惡意破解

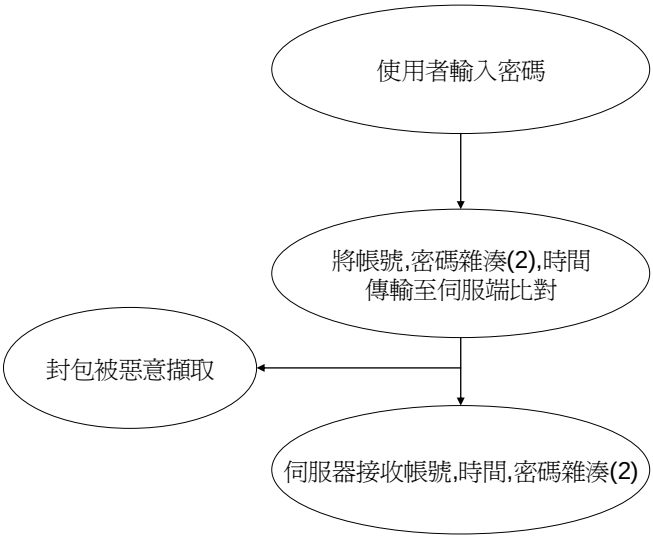

圖 3-7 SHA-1 圖形流程簡介（5）

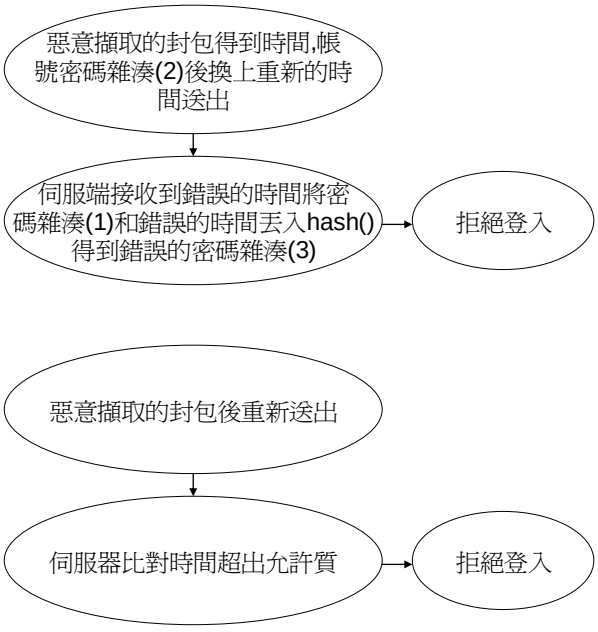

圖 3-8 SHA-1 圖形流程簡介（6）

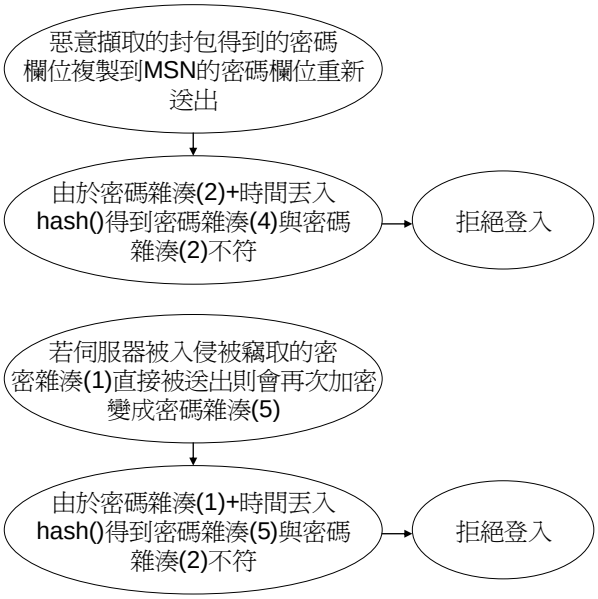

圖 3-9 SHA-1 圖形流程簡介（7）

1. 何謂SHA-1

安全雜湊演算法(Secure Hash Algorithm)簡稱SHA，是由美國國家標準與技術協會(NIST)所發展出來的，並且在1993 年發布成為第180項美國聯邦資訊處理標準(FIPS PUB 180)，而在1995 年底又將其修訂版本發布為FIPS PUB 180-1，所以我們通稱此版本為SHA-1，SHA是以MD4(Message Digest Algorithm)為基礎，並且設計方式與MD4也很類似。

2. SHA-1 原理

SHA-1 演算法所輸入的訊息不長度不能超過2^{64}個位元，而輸出則

是一個160 位元的訊息摘要，輸入的訊息會被分成好幾個512 位元的區段來處理， SHA-1 的處理流程可以由圖5-1 來了解，其區段長度為512位元，雜湊碼的長度與串接變數長度都是160 位元。

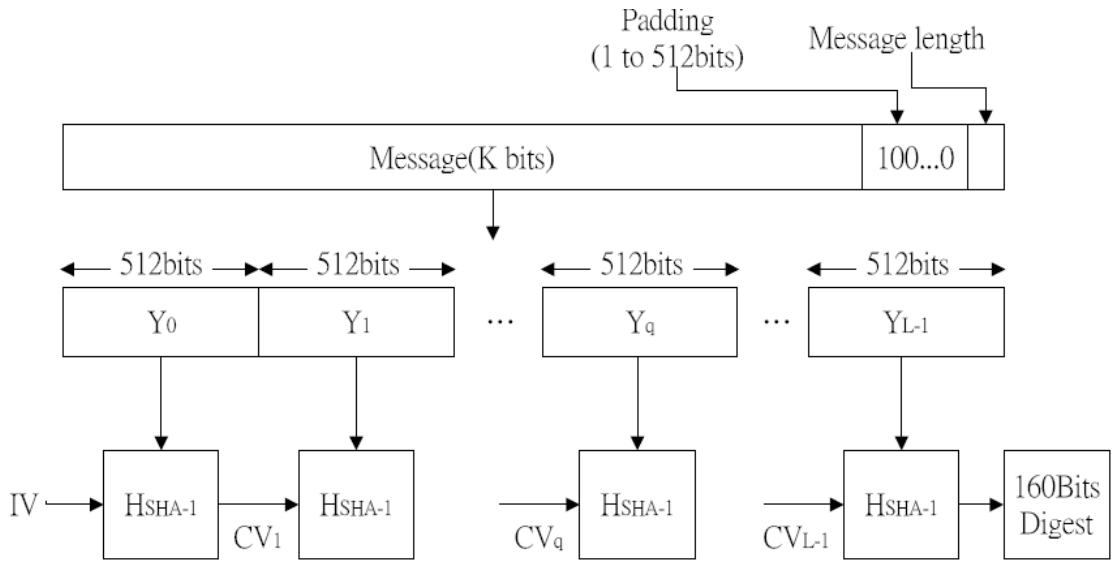

圖 3-10 SHA-1 示意圖

3. SHA-1 訊息附加位元(Padding bits)

在訊息之後附加一些位元使訊息取512 同餘之後等於448，我們一定要在訊息尾端附加位元即使訊息本身長度就已經符合我們需求，因此我們附加的位元個數可以從1 到512 位元，而附加的方法是先加上一個1 然後再用0 補到需要的長度，當然這需要在訊息末端加上一段64 位元的資料，這段資料以一非負整數表示用來記錄原來訊息的長度。

4. 設定Message Digest 暫存區的初值

我們使用一個160 位元的暫存區來存放這個雜湊函數的中間值及最後結果，我們可以用5 個32 位元暫存器(A、B、C、D、E)來表示這個暫存區，而這5 個暫存器的起始初值如下(以16 進位表示)：

A = 67452301

B = EFCDAB89

C = 98BADCFE

D = 10325476

E = C3D2E1F0

圖 3-11 SHA-1 暫存示意圖

5. 處理訊息中512 位元區段

此為SHA-1 核心部分，這部分是由四個「處理回合」所組成的模組，每個回合有20 個步驟，運作邏輯我們可以由圖5-2 看到，這四個回合的結構都差不多，但是每個回合都用了一個不同的基本邏輯函數，這些邏輯函數在這個規格中分別被標示成F1、F2、F3 及F4。

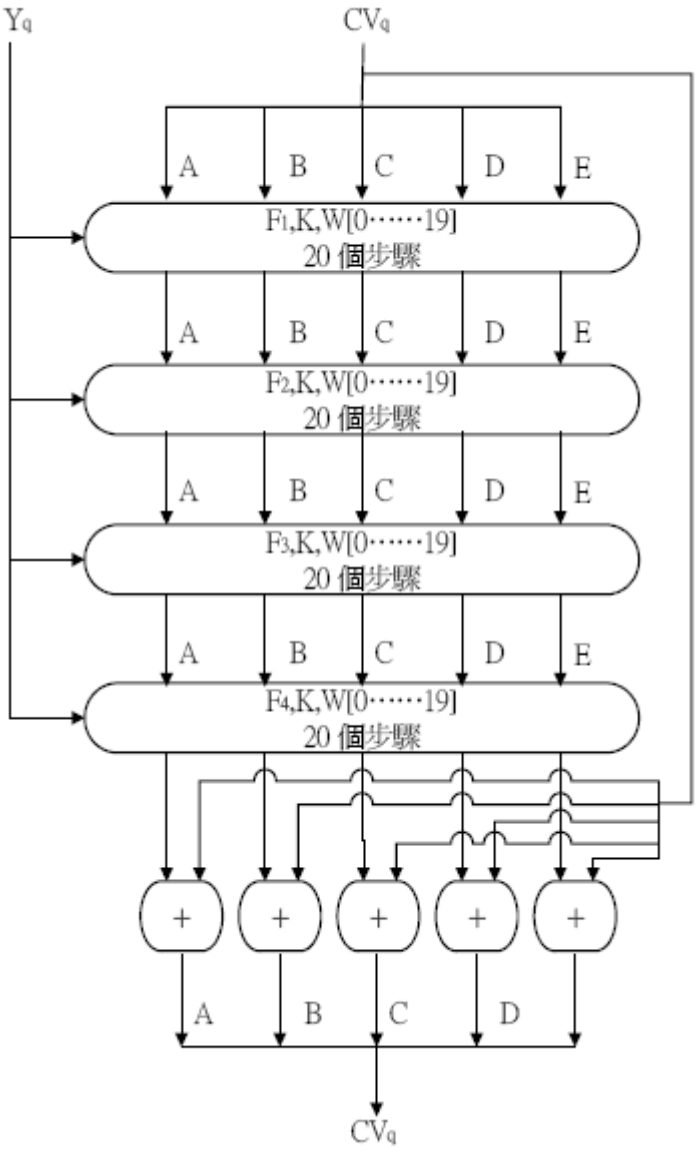

圖 3-12 訊息處理流程

每回合的輸入是我們正要處理的512 位元區段(Yq)與160 位元的暫存區ABCDE，這四個回合會分別更動這個暫存區的內容，每個回合會加上常數Kt，我們用步驟編號t 由0 到79 來表示四個回合中的80 個步驟，而實際上我們只用了四個不同的常數如下所示：

K_t = 5A827999 $(0 \le t \le 19)$

K_t = 6ED9EBA1 $(20 \le t \le 39)$

K_t = 8F1BBCDC $(40 \le t \le 59)$

K_t = CA62C1D6 $(60 \le t \le 79)$

圖 3-13 訊息暫存示意圖

第四個回合的輸出(第80 個步驟的輸入)會跟第一個回合的輸入(CVq)加在一起，所產生的結果就是CVq+1，相加的方法是暫存區中的四個字元與CVq 中相對應的字元相加(字元間彼此獨立相加)，並且都要取232的同餘就可以得到結果。

6. SHA-1 輸出

當所有512 位元的區段都處理過之後，最後一階段產生的輸出就是我們要的160 位元的訊息摘要值。我們可以將SHA-1 的行為歸納如下：

$$CV_0 = IV$$

$$CV_{q+1} = SUM_{32}(CV_q， ABCDE_q)$$

$$MD = CV_L$$

此處

IV = ABCDE 暫存區的初始值

$ABCDE_q$ = 訊息中第q 個區段最後一回合的輸出

L = 訊息中的區段個數（包含以附加的位元及長度欄）位

SUM_{32} = 將兩個輸入區段的字元分別獨立相加之再取2_{32} 的同餘

MD = 最後的訊息摘要值

7. SHA-1 的壓縮函數

仔細來看SHA-1 中每回合的每個步驟中是如何處理一個512 位元的區段，每一個步驟都具有下列的形式：A,B,C,D,E←（E + F（t，B，C，D） + S5（A） + Wt + Kt），A， S30（B），C，D每個基本函數都需要三個32 位元的輸入字元，然後會產生一個32 位元的輸出字元，每個函數都會執行一組位元對位元的邏輯運算，換言之輸出的第n 個位元是由三個輸入的第n 個位元所產生的。這些函數可以歸納如下所示：

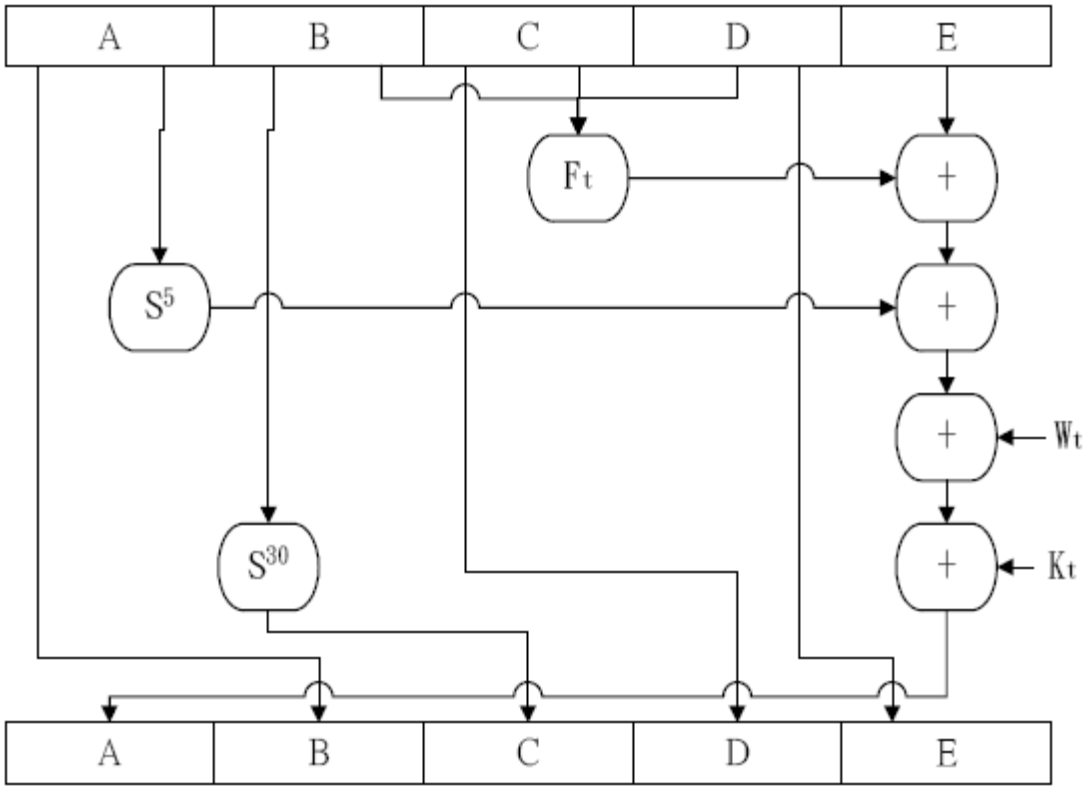

圖 3-14 壓縮流程示意圖

8. SHA-1 和MD5 的比較

為什麼我們要採用SHA-1 而非大家慣用的MD5，SHA-1 和MD5 都是從MD4 衍生而來的，所以兩者之間有許多相似之處。因此他們的強度和特性也很相似，我們就根據MD4 所定義的設計目標來比較這兩個演算法。

對抗暴力法攻擊的安全程度：最明顯也是最重要的差別就是SHA-1 的摘要值要比MD5 的摘要值多了32 個位元，如果我們想針對一摘要值來求出訊息的話，MD5 需要執行2^{128} 個運算，而SHA-1 則需要執

行2^{160} 個運算。此外如果想要找到兩個可以產生相同摘要值得訊息的話，那MD5 需要執行約2^{64} 個運算，而SHA-1 則需要執行約280個運算，因此就暴力破解法而言SHA-1 是比較安全的。

對抗密碼破解的安全程度：近年來MD5 在結構上被人發現是不安全的，而SHA-1 尚未有發現結構上的問題，加上SHA-1 的設計策略和DES 的S-box 相同不為人知，所以目前看來比MD5 要安全許多。

速度：這兩個演算法都大量的使用了取232 同餘的加法，所以他們在32 位元處理器上的表現都很好，但是SHA-1 需要的步驟是80 步而MD5 是64 步，在暫存區的長度上SHA-1 的160 位元也比MD5 的128 位元要來的多，所以在同樣的電腦上SHA-1 的執行速度要比MD5 來的慢。

由上面幾點來看雖然SHA-1 的執行速度較MD5 為慢，但是SHA-1 卻具有較高的安全性強度，加上MD5 被發現有結構上的缺點，所以我們當然是採用SHA-1。

第四章　系統呈現

第一節　預期效能與實際效能的比較

(一) 圖形的傳輸

此系統我們已經有建立圖形傳輸功能,但是因為圖形傳輸在 Swing 領域裡屬較深入之技術,而我們此系統主要以 Swing、SQL、NET、Screen 這四種技術領域並重，因圖形傳輸技術太過深入會進而拖慢了其他 3 種技術的進度，所以我們並無設計的很完善。

(二) 遊戲

遊戲系統的設計需要用到 Flash 系統，而 Flash 遊戲的製作要花很多時間去設計遊戲介面，假如進行製作的話，則會拖到其他系統的進度故我們先暫時不進行。

(三) 圖形介面

因為我們不是美術相關科系出生，假如在去製圖的話可能會花相當多的時間在這部份，所以這部分我們沒有去設計，因為我們主要是學術研究，所探討的是技術層面的東西。

第二節　系統畫面

以下三張是系統登入勿畫面

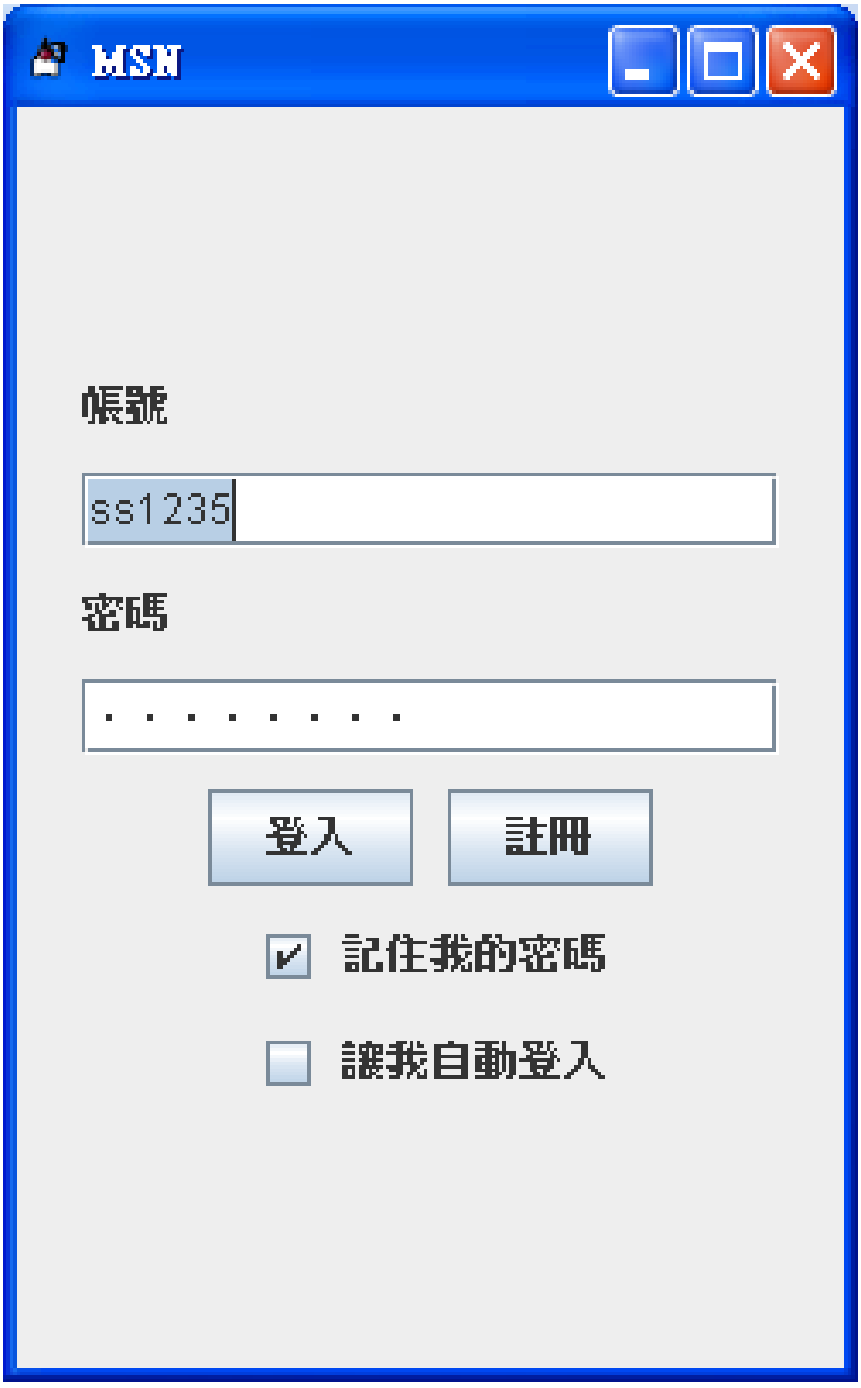

圖 4-1　記憶帳號密碼

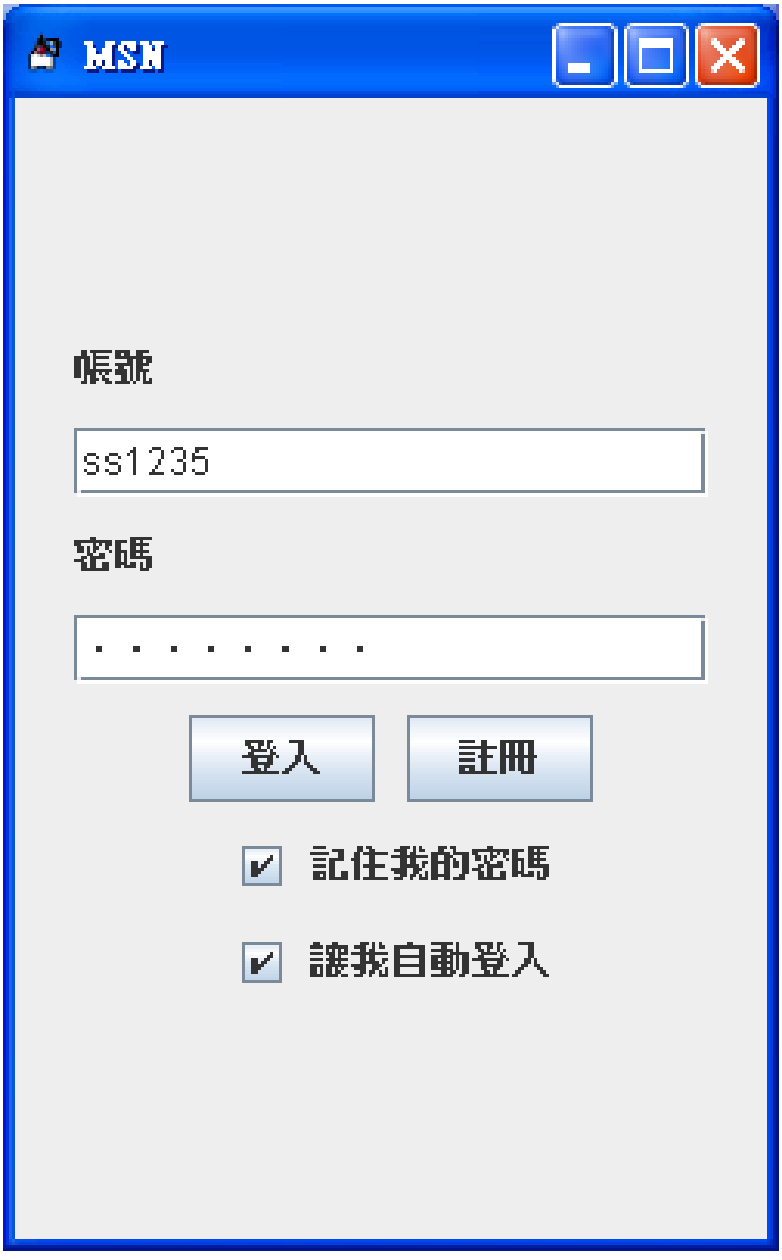

圖 4-2 自動登入

圖 4-3 登入中

畫面登入後，顯示好友上線

圖 4-4 登入後

圖 4-5 好友上線

接下來幾張是更換圖片的視窗

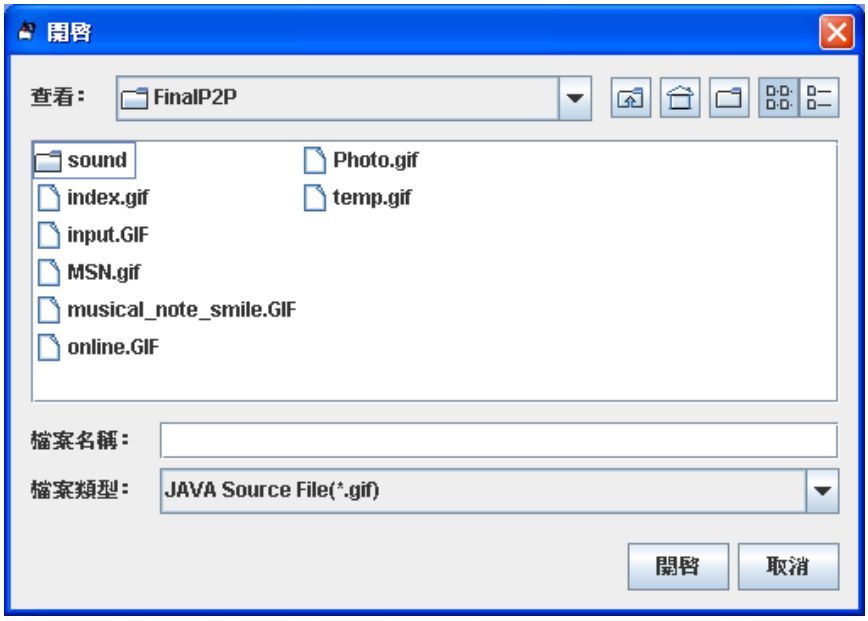

圖 4-6 更換相片

圖 4-7 更換相片後

圖 4-8 好友登入後

以下二張是暱稱狀態的變換

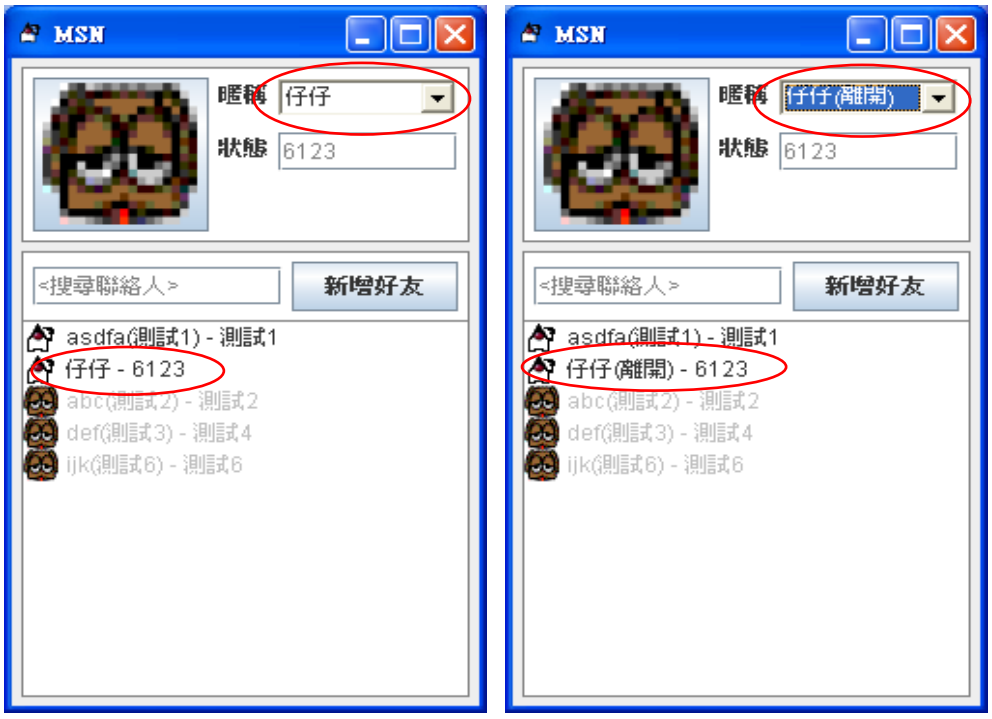

圖 4-9 改變暱稱

圖 4-10 改變狀態

傳送訊息的視窗

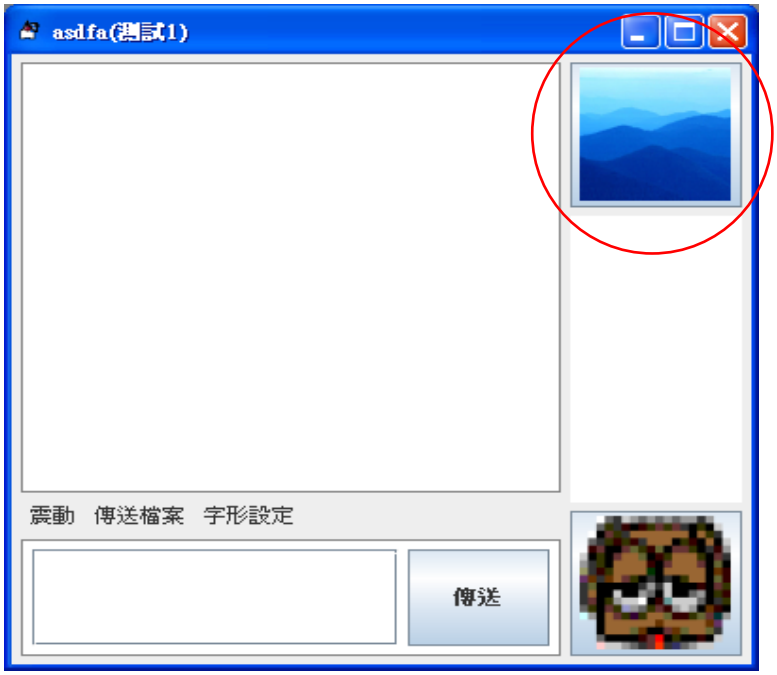

圖 4-11 好友視窗

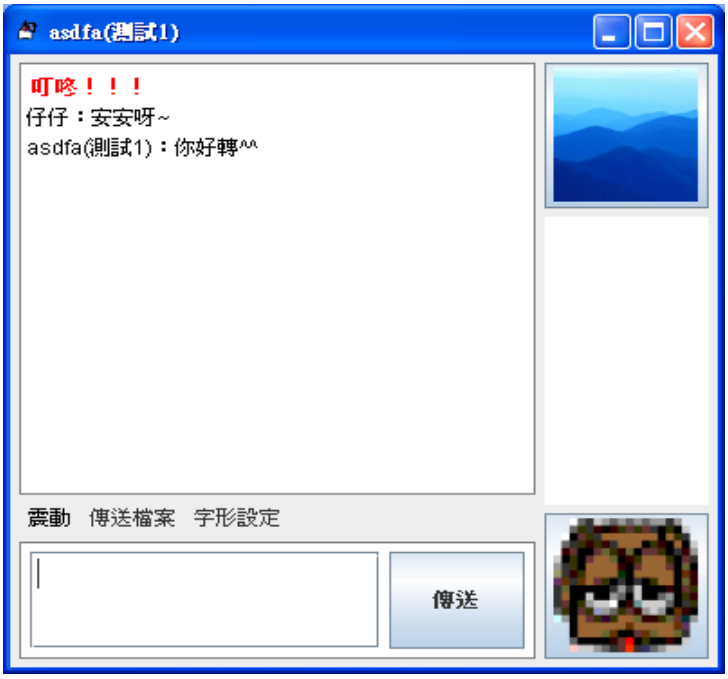

圖 4-12 傳送訊息

改變字體顏色的設定

圖 4-13 字型設定

發出訊號讓對方收到

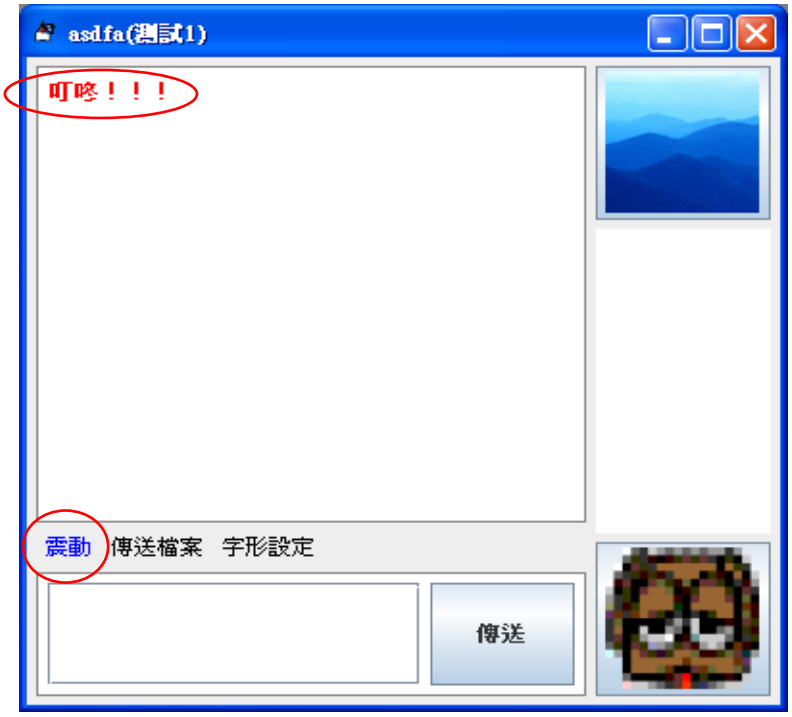

圖 4-14 發出震動

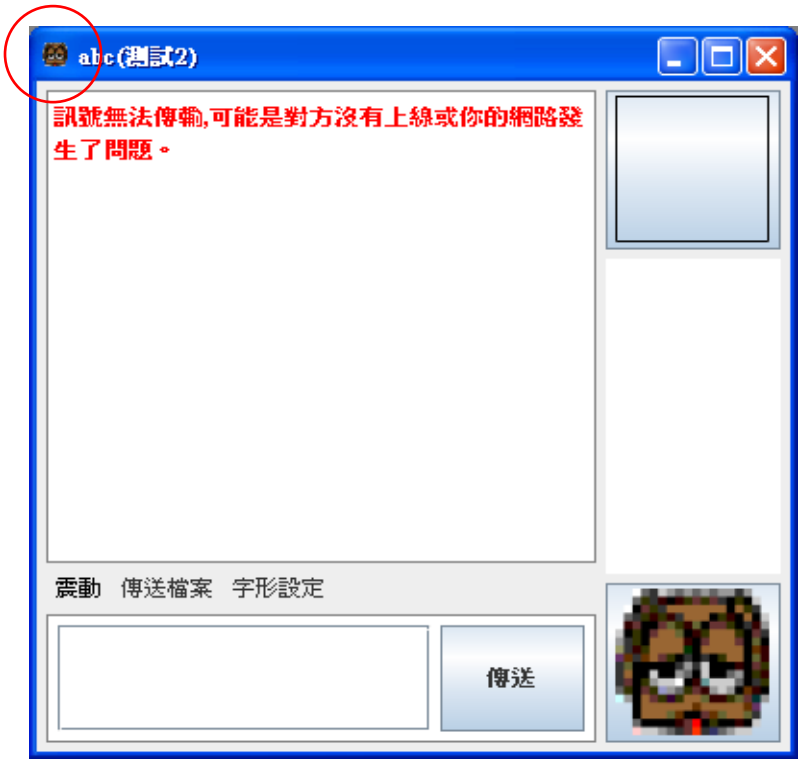

圖 4-15 傳送離線訊息

帳號註冊

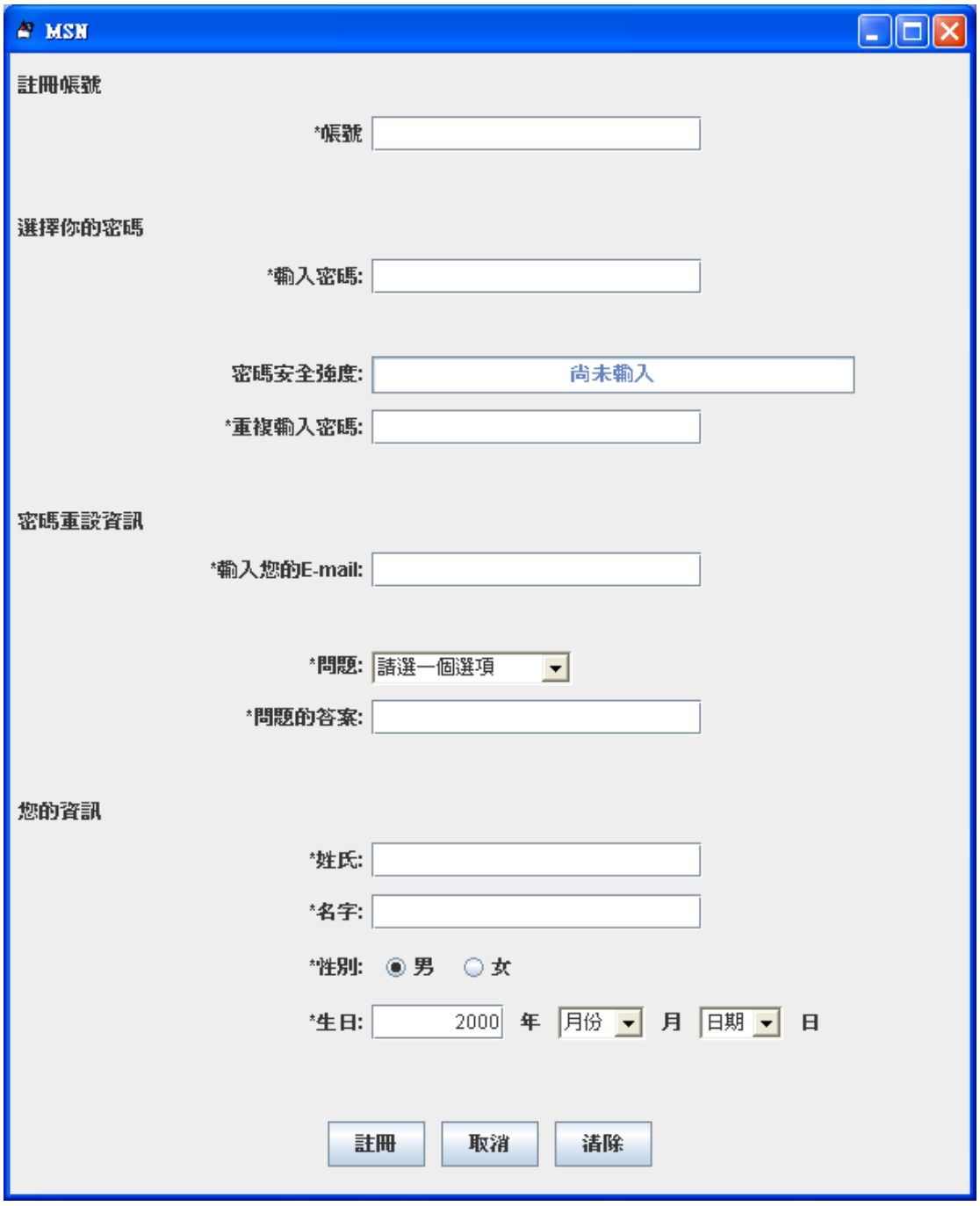

圖 4-16 註冊畫面

確認註冊者所輸入的資料是否正確

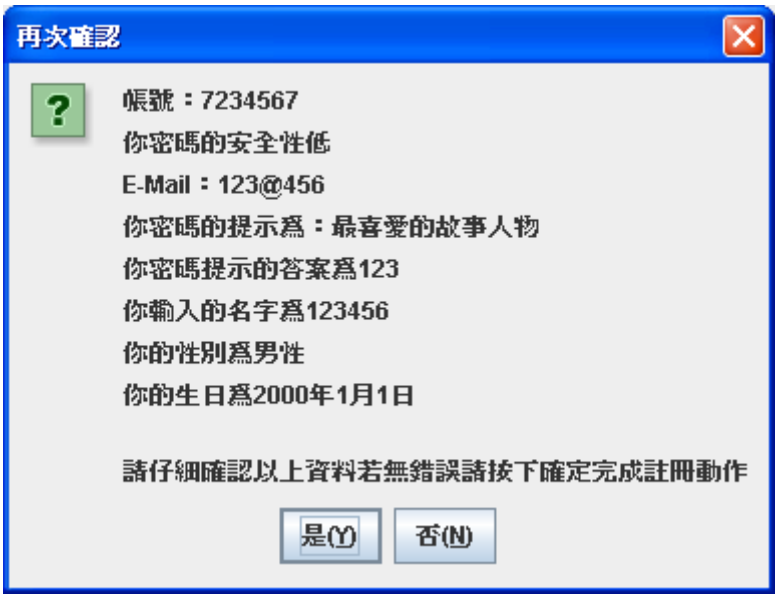

圖 4-17 註冊確認

第五章　結論與未來展望

第一節　結論

本研究以 Java 來實作資料傳輸系統，由於 Java 語言讓程式設計者與開發者能夠建立穩定、可攜、與跨平台的產品，其使用範圍日漸普遍。而我們所做的資料傳輸系統，就是網際網路，在網路上傳遞各項資料，當連接時，此通道同時為使用者提供全雙工、即時的資料傳輸。我們的資料傳輸系統以點對點的方式傳輸，並且支援多點傳送，更可以說是善用了 P2P 的特性。

網際網路與全球資訊網的相關技術的出現未我們的生活帶來相當戲劇性且重要的改變。在生活上許多應用程式中，資料傳輸已經備受矚目，並成唯一種新興的社交活動。無論是傳統的網路業者或是即將跨入網路市場的新興業者，提供網路雙向傳遞的服務是相當重要且必要的。

我們此專題最重要的特色就是全雙工資料傳遞，而且使用者除了須付網路的月租費之外，此軟體系統完全免費，以及就是可以群體資料傳遞與接收。

第二節　未來展望

城市未來的發展性跟改進的方面可以達到有效的發送與接收再同一個 Java 視窗上的整合，也就是所謂的介面圖形化。使我們的程式更加的人性化，並增加線上選擇聲音取樣率、語音留言、會議交談...等系統，使我們程式可以更方便進而取代現實中類似相同的程式系統。

參考文獻

參考網站

http://zh.wikipedia.org/wiki/SHA_%E5%AE%B6%E6%97%8F

http://www.nii.org.tw/cnt/info/Report/20020901_1.htm

http://www.pcnet.idv.tw/pcnet/network/network_ip_tcp.htm

http://www.freebsd.org.hk/html/solaris8/TCP&UDP.html

http://tw.knowledge.yahoo.com/question/?qid=1305092813092

http://tw.knowledge.yahoo.com/question/question?qid=1306061810567

http://zh.wikipedia.org/wiki/Microsoft_Access

http://www.digitimes.com.tw/ext/ext.asp?BigExtID=457

http://infotrip.ncl.edu.tw/net/tcpintro.html

參考論文

1. 林建興，**2006.6**，個人軟體流程改善之協定設計 與電腦輔助工具之開發，逢甲大學資訊電機工程碩士在職專班碩士論文。

2. 李俊嶧，**2005.6**，分散式視訊會議服務之設計與實作，國立中山大學資訊工程學系碩士論文。

3. 洪人傑，**2007.7**，**P2P** 網路去污染檔案容錯機制之研究，國立成功大學工程科學研究所碩士論文。

4. Jacky Lee，2005.3，Eclipse-整合開發工具初級篇