逢 甲 大 學

資訊工程學系專題報告

XML Mail安全傳送

學 生： 陳 威 宇 (四乙)
陳 弘 奇 (四乙)
黃 政 勳 (四乙)

指導教授 ： 李維斌

中華民國九十二年十二月

目錄

圖表目錄：

摘要

本專題是用來時作如何將email轉換成xml，並分析其特定欄位的值成為輸入，使得本系統能全自動的判斷是否要做多工的處理，如壓縮、加密，而本系統也考慮到傳送失敗和傳送中斷等連線的情形，因此有續傳以及續傳失敗換第二伺服器傳送等功能。

本組依循專案開發的基本步驟，透過系統分析、評估、實作以至解決問題，來完成此專題及報告書。

本報告架構共分六章，撰寫方式是依循開發過程的順序，並加入了圖表輔助說明。第一章為導論，說明本專題的基本概念及我們專題小組的工作情形，第二章為系統概念，設計本系統所需要的背景知識皆收錄於這個章節。

第三章之後開始進入主題。第三章從IT的角度來作本專題的系統分析與設計，還有安裝系統及效能評估，第四章就是整個實作的部分，包含了整個系統各部分間我們是如何設計，第五章是我們在做專題時遇到的困難及如何克服，第六章則是我們系統的展望及每個人的心得。

第一章導論

1.1 研究動機與目的

大三的時候, 開始主修於網路概論的相關課程, 那時對於網路的我們, 專題想專注於網路傳輸及應用, 因此我們開始思考題目時, 專題指導老師剛好有一個關於網路的轉換傳輸, 於是我們的專題開始啟程。隨著資訊化時代的來臨，人們對於網路的需求明顯增加，不論是個人的資料傳遞，或是私人公司的商業資料，將有愈來愈多的機會是透過網路作為媒介來傳輸的，所以一套具有強大功能的私人傳輸系統, 來保護資料達成私人機密的傳輸。

由於 XML 是一項新技術, 我們把它和電子郵件的轉換套用, 是一項新的嘗試, 由於技術上面成熟, 對這些功能定義適當的 TAG 必須重新的去定義, 讓電子郵件的功能擴增, 也能享用 XML 的優點, 包括 可驗證文件內容的標準化文法、區分文件結構內容和呈現方式等功能, 使得在使用者和應用程式之間交換文件更為容易。

1.2 何謂資訊安全閘道 XML 轉換傳送接收

這是一個網路傳輸系統, 分為 client 端和 server 端。當 client 端時, 它擁有一個目錄, 會隨時的偵測和掃描目錄中是否有新郵件, 當有新郵件時, 他會自動的依據設定檔的設定值, 去尋找的目的地的 IP 位址, 然後將郵件送過去目的地; 當郵件要輸送之前, 依據郵件中的 Tag 去適當轉換成 XML 文件, 之後再把 XML 文件 DES 加密, 形成一個加密的 XML 文件, 而後加以壓縮分割檔案搭配 JAVA Socket 的傳輸, 把郵件有效的傳送過去; 而 client 端也有支援續傳的原理, 讓不完全的檔案, 可以重新啟動傳送, 不用全部重新傳送; 也有 IP 優先權的選擇, 可以不受故障的網路的影響, 選擇其他優先 IP, 支援傳送。而 server 端的部分, 它會等待郵件的傳送輸入, 當有郵件進入時, 可以馬上把郵件檔案解壓合併檔案, 而後解密 XML 文件, 在把 XML 轉成電子郵件檔。

1.3 系統簡介

- Java 2 SDK Standard Edition(J2SE 1.4)
- Java Cryptography Extension(JCE 1.2)
- Java API for XML Processing(JAXP 1.1)
- Cryptix
- IAIK

此專題報告是用於資訊安全閘道設計,程式是用 Java 開發而成,適用於電腦的各種平台,可搭配掃毒、監控資料庫等程式,變成一個完整的安全閘道。隨著網際網路的日漸普及，資訊的傳遞普遍利用網路來進行，資料的傳輸也都是用網路來輸送,對於網路的網路的傳輸中斷和加密的所求也日漸需要和重視。

本系統的最終目的就是用於私人的網路傳安全傳輸,對於私人企業的機密資料如果需要網路傳輸,就可以搭配此系統,行程自己的一個安全的網路傳輸路線。資訊安全是進行網路傳輸的首要考量,本系統提供相關的加／ 解密技術。系統使用 XML 用來傳遞這些資料的最好方式,雖然 XML 剛發展不久,但靠者 XML 強大方便的特性可應用應用在政府的公文交換或是電子商務上。在於網路的中斷處理,系統也能隨時的偵測,給予最大的善後處理,讓資料完整的擁有。

開發語言:

專題實驗主要是用與 Java 來完成，而大部分都是使用一些 Java 套件,來對各種程式撰寫。XML 部分使用 JAXP API 來支援 XML 的 DOM、XSLT、SAX 的格式,讓 Java 輕鬆的寫出 XML 文件。加解密是搭配 JCE、Cryptix、IAIK,再使用 DES 演算法完成加解密工作。Server 與 Client 溝通是使用 Java 的 Socket 來寫成的。

開發環境:

安裝任何平台上,直接於任何平台上測試,是開發本專題的重要環境,而大致上就是一些 JCE、JAXP 的安裝和設定及 Java 的 API 套件,就可以開發程式此系統的架構。

1.4 成果簡介

	陳威宇	黃政勳	陳弘奇
系統開始分析與建置			
系統架構討論	●	●	●
資料收集	●	●	●
系統分析	●	●	●
規劃系統	●	●	●
DES加密研究	●		
XML轉換研究	●(SAX)		●(DOM)
續傳方法研究	●	●	●
Server 端程式功能區塊			
接收檔案、續傳		●	●
組合分割檔	●		●
解壓縮、解密	●	●	
轉信件格式			●
傳送失敗檔案判斷、清除		●	

Client 端程式功能區塊			
讀取信件			●
XML格式、處理			●
加密、壓縮	●	●	
傳送檔案、續傳	●	●	●
系統整合 — Client 和 Server 端			
Client端		●	●
Server端	●	●	●
介面設計	●		
執行環境研究 — XML & 加密JCE & 測試			
JCE加密環境	●	●	
XML環境	●		●
系統測試	●		●
書面文件、報告			
操作文件	●	●	●
書面報告	●	●	●

表 1-1 工作分配表

月 份	工 作 編 號	內 容
2 月 ~ 3 月	T1	資料收集
3 月 ~ 4 月	T2	系統分析
4 月 ~ 5 月	T3	演算法及系統可行性評估
5 月 ~ 6 月	T4	Server 端程式撰寫及測試
6 月 ~ 7 月	T5	Client 端程式撰寫及測試
7 月 ~ 8 月	T6	介面修飾系統測試
8 月 ~ 9 月	T7	系統整合系統測試修改
9 月 ~ 10 月	T8	系統整合
10 月 ~ 11 月	T9	報告編排及撰寫
11 月 ~ 12 月	T10	書面報告

表 1-2 進度分配表

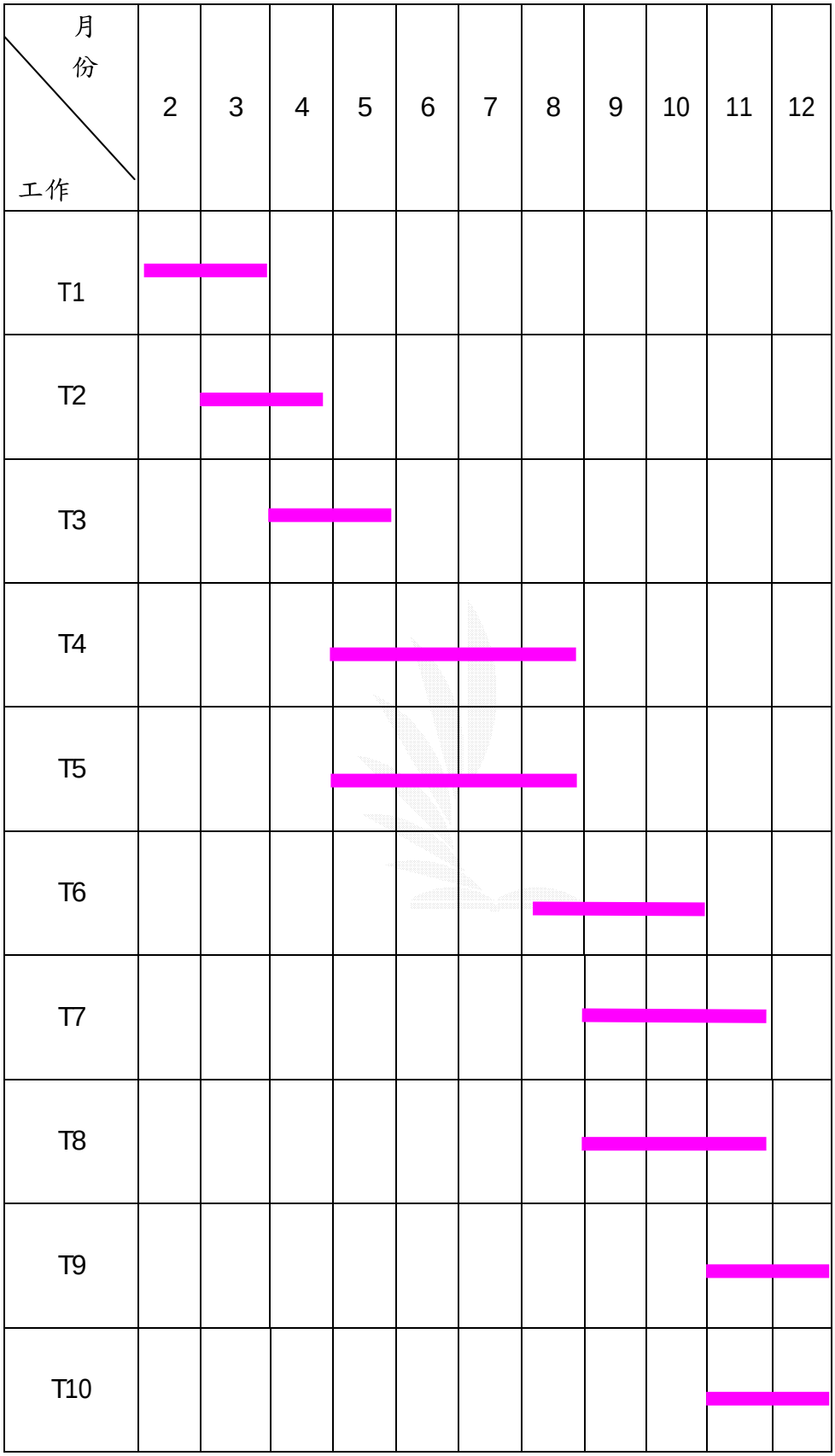


圖 1-1 工作進度甘特圖

第二章系統概念

2.1 JAVA 相關簡介

2.1.1 JAVA 介紹

Java 是由 Sun 公司所發展出來的程式語言，中文譯名為爪哇，它本身是一種物件導向(Object-Oriented)的程式語言。Java 也號稱是能跨平台使用的語言，這主要是因為 Java 本身被編譯之後，並不是直接產生可執行的碼，而是產生一種中間碼叫作 ByteCode，這種碼必需在透過 Java 的直譯 器來解讀它，才能夠真正的被執行，所以只要平台上裝有這種 Java 的直譯器，就能解讀 ByteCode 也就能執行 Java 編譯過的程式，故與 Java 程式是在那種平台上被編譯的，就完全沒有干係了, 因此適用於任何平台的作業系統。

2.1.2 JAVA 平台環境設定

要在 FreeBSD 上 compile Java 程式有兩種做法:

1. 用 kaffe + JDK.

Kaffe 是一個 Java 的 VM 可以用來執行 Java 的 byte code. 若要 compile 程式則還要拿 JDK 中的 class.zip 來用. 如果是自己拿 kaffe 的 source code 來 compile 的人應該不用我多說甚麼, 如果是要 package 來安裝的, 可參照下面的步奏:

1. pkg_add jdk-1.0.2.tgz
2. pkg_add kaffe-0.8.2.tgz
3. 如果是用 csh/tcsh, 在 .cshrc/.tcshrc 中加入

setenv CLASSPATH /usr/local/share/java/classes.zip:\

/usr/local/share/kaffe/biss.zip:.
setenv KAFFEHOME /usr/local/share/kaffe
setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH:/usr/local/lib
set path=($path /usr/local/bin)

做完後 source 一下 .csrhc/.tcshrc 就可以用 javac, javadoc, kaffe, kaffeh 了. 如果是用 sh/ksh/bash 自己應該知道怎麼辦吧! 如果不知道 JDK 和 Kaffe 怎麼用, 請自己看 JDK 的文件和 kaffe 的 source code 中的說明.

2. 用 guavac, 自己去找 guavac 的 source, 或用 package 中的 binary 皆可. 以目前的 compile 速度而言, guavac 快了許多, 若要執行 java byte code 的話, 那麼 guavac 就使不上力了.

2.1.3 JAVA 特點

1. 簡單(Simple)：容易撰寫程式，不需要長時間的訓練，而能滿足現代的需求。程式小型亦是簡單的一種特性，使得軟體能夠在小型機器上執行,基本的解譯器約為 40k,若加上基本的程式庫,約為 215k。

2. 物件導向的(Object-Oriented)：物件導的設計是一種重心在資料和介面的技巧。若以木工為比喻，一個以物件為導向的木工，他(她)最主要的重點是即將要做的木椅子,其次才是所需要的工具;反之;一個以非物件為導向的木工，他(她)所關心的只是工具。最近的隨插即用(Plug and Play)亦是物件導向設計的重點。

3. 分散式的(Distributed)：Java 有一個很週全的程式庫，且很容易地與 HTTP 和 FTP 等 TCP/IP 通訊協定相配合。Java 應用程式(Applications)能在網路上開啟及連結使用物件，就如同透過 URLs 連結使用一個區域檔案系統(Local File System)。

4. 強韌性的(Robust)：由 Java 所撰寫出的程式能在多種情況下執行而

具有其穩定性 Java 與 C/C++最大不同點是 Java 有一個指標器模型(Pointer Model)來排除記憶體被蓋寫(Overwriting Memory)和資料毀損(Corrupting Data)的可能性。

5.安全性的(Secure)：Java 是被設計用於網路及分散性的環境中，安全性自必是一個很重要的考慮。Java 擁有數個階層的互鎖(Interlocking)保護措施，能有效地防止病毒的侵入和破壞行為的發生。

6.架構中立性的(Architecture Neutral)：一般而言，網路是由很多不同機型的機器所組合而成的，CPU 和作業系統架構均有所不同；因此，如何使一個應用程式可以在每一種機器上執行，是一個難題。所幸，Java 可以應用上.

2.1.4 JAVA 程式設計與說明

1. 類別:

類別名稱必須以一個大寫字母開頭（例如，string），如果類別名稱包含了一個以上的字， 則每個字的開頭字母都必須為大寫（ 例如，StringBuffer）。如果類別名稱或是類別名稱中的某個字，即首字母縮略字（acronym），則該首字母縮略字的字母必須全部使用大寫（例如：url、HTMLParser）。介面名稱（ 例如：docment、FileNameMap、Collection）。

2. 方法:

方法的名稱必須永遠以一個小寫字母開頭，如果名稱中包含了兩個以上的字， 則從第二個字開始， 每個字的第一個字母都必須為大寫（ 例如：insert ( )、insertObject ( )、insertObjectAt ( )）。典型的方法名稱第一個字都會使用動詞，同時會將必須的字納入，以讓該方法的用意夠明確清楚，但會盡量選擇簡短的字。

3.欄位與常數:

非常數欄位的名稱所遵循的規則與方法的名稱相同。如果欄位是個ｓｔａｔｉｃｆｉｎａｌ常數，則它必須使用大寫來表示。如果常數名稱包含了二個以上的字，則字與字間必須使用底線來加以隔開（例如：ＭＡＸ＿ＶＡＬＵＥ)。欄位名稱必須盡量選擇能夠清楚表達該欄位值所代表的涵意。

4.參數:

方法參數的名稱出現在該方法的說明文件（documentati on）中，因此你必須選擇能夠清楚表示出該參數所代表的意義的名稱。盡量讓參數名稱只含有一個字，並前後一致地使用它們。例如，如果 WidgetProcessor 類別定義了許多方法，其以一個 widget 物件作為第一個參數，則你必須在每個方法中將此參數命名為 widget 甚至也可以是 w。

5.區域變數:

區域變數的名稱是實作上的細節，其實在你的類別之外也見不到它們。然而，選擇適當的名稱會讓你的程式碼更加地易於閱讀、了解以及維護。典型的變數名稱會遵循方法與欄位名稱的命名慣例。
除了對上述幾種名稱的命名慣例之外，有許多關於字元的慣例也是你必須遵守的。Java 允許$ 出現在任何的識別子（i dent if i er）中，但依據慣例，它是被保留下來供作原始碼處理器所產生的虛構名稱之用（例如，Java 編譯器會使用此字元來運作內部類別）。

2.1.5 JAVA 網路

2.1.5.1 JAVA TCP/IP 通訊協定

Java 程式設計中，其實不需要擔憂 IP 的內部作業細節，但你必須知道 IP 定址系統。在 IP 網路上，每部電腦都是由一個四位元組（３２- ｂｉｔ）的號碼來代表，這也就是你常聽到的 IP 位址，一般你看到的 IP 位址應該是類似 192.168.100.3 這樣的格式，每個號碼的長度都只有一個位元組，也就是每個值都是介於０到２５５之間。每部連上 IP 網路的電腦都有一個獨一無二的四位元組位址。當資料經由網路傳遞時，封包的標頭會包含來源主機與目的主機的位址；路由器會檢查封包內目的主機之位址，然後找出最佳的路徑把封包引導到目的地。因為封包內有包含來源位址，所以接收端會知道到底是誰送出來的資料，並知道該對誰回應。電腦是處理數字的高手，但是人類卻不擅長記住它們，因此必須要有一套方式把數字（ 像是１９２．１６８．１００．３）轉換成人類方便記憶的網址（ 像是ｗｗｗ．ｏｒｅｉｌｌｙ．ｃｏｍ．ｔｗ），於是網域名稱系統（Domain Name Syst em，DNS）就因運而生。當然，轉換是雙向的，若需要把文字網址對應到數字 IP 位址，一樣可以利用 DNS 來辦到。當 Java 程式運用網路時，也必須處理這些數字位址與主機名稱之間的轉換，在 inetaddress 類別中，有一堆函式可以處理這件事。

2.1.5.2 java java.net 套件

Java 網路程式使用 java.net 套件類別的網路 I/O,可分為兩種層次:

1.高階的 URL 類別;

URL 類別的物件是 WWW(World Wide Web)的 URL 網址, 全名是萬用資料定址器(Universial Resource Locator),這各類別提供多種方法剖

析 URL 網址, 提供 Web 的 HTTP(HyperText Transfer Protocol)的連線功能。

2.低階的 Socket 類別:

Socket 類別可以建立網路間的連線, 以便 Java 程式透過網路連線到其他的 Java 程式或 Internet 服務, 建立' 主從架構'(Client/Server)的網路應用程式。

2.1.5.3 JAVA ServerSocket 簡介及應用

ServerSocket 類別 : 提供 server 端網路程式等待 client 端網路程式連線的功能. 並且向作業系統要求一個 socket 資源使用之前必須 import java.net.ServerSocket;

server 端網路程式例:

IP address 127.0.0.1 指的就是電腦自己 ServerSocket listen_socket = new ServerSocket(20353); 向作業系統要求 port number 20353 做為傾聽用(client 的連線) telnet 163.18.62.211 20353 用 telent.exe 這支程式向 163.18.62.211 之 port number 20353 提出連線要求而(163.18.62.211 , 20353)這個 srever socket 是由 TESTserver 這支 server 端網路程式所建立 listen_socket 是一個 ServerSocket 物件 while(true) {

```
Socket client = listen_socket.accept();
// ServerSocket 物件有一個 accept()函式會等待 client 端連線
// 進來, 若有 client 連線, 此函式會回傳一個 Socket 物件, 例如
// client. 如果沒有連線進來,accept()就一直 waiting.
// 意思是程式停在這裡而不往下執行
PrintStream stoc = new PrintStream(client.getOutputStream());
// 為客戶端建立一輸出資料流
stoc.println("This message comes from dog server");
client.close();
```

// 結束連線}
client.getOutputStream()
client 是一個 Socket 物件,getOutputStream()會建立一個 Output stream PrintStream stoc = new PrintStream(client.getOutputStream());
把 I/Ostream 包裝成更高階的 I/O, 例如 PrintStream stoc
stoc.println("This message comes from dog server");
透過高階 I/O PrintStream 物件 stoc 的函式, 我們就可以很方便把字串 "This message comes from dog server"送往 OutputStream 因為該 OutputStream 是 client 的連線, 所以字串就"流"到 client 端.

2.2 XML 相關簡介

2.2.1 XML 介紹

XML 是一九八六年國際標準組織(International Standards Organization, ISO)公佈的一個名為「標準通用標示語言」(Standard Generalized Markup Language, SGML)的精簡版/子集合。一九九八年二月，美國 W3C 組織正式公佈 XML 的 Recommendation 1.0 版語法標準。XML 掌握了 SGML 延展性、文件自我描述特性、以及其強大的文件結構化功能，但摒除了 SGML 過於龐大複雜以及不易普及化的缺點。換言之，XML 是一種用來定義其它語言的語法系統。這正是 XML 功能強大的主因。它可促進各專業機構、不同產業界、學術界和特定應用領域發展各自標準的文件和訊息， 以利資訊的交換、處理和相關衍生性資料加值服務。

XML 文件和訊息的主要特色在於它是結構以及資訊內容導向。結構化紋健和訊息編瑪方法的主要精神在於它可供其它電子資料傳遞、文件出版系統、電腦輔助設計或製造、資料庫管理等系統， 在處理重複和共享的資料時， 能有效提升其效率和效能， 節制資訊系統的開發建置和管理營運成本。維的可能性。

2.2.2 XML 的優勢

1.XML 會啟用可延伸的標示集，以提供更精確的文件內容說明。

XML 實作者可以定義本身的標示集來說明文件內容。這些說明的精確度是由實作者來決定。比方說，某項實作可能會使用 <name> 標示，而另一項實作可能會發現 <city_name> 標示更有用。這個功能優於 HTML；因為使用 HTML 時，開發人員只有一組固定的標示集可供選擇。

2.XML 可驗證文件內容的標準化文法。

XML 文件的內容和結構由它的文法來定義。「文件類型定義 (DTD)」是這種文法的一個範例。其它 XML 型的綱目仍在發展中。在這份文件中，我們使用文法來通稱這些綱目。

3.XML 使得在使用者和應用程式之間交換文件更為容易。

從早期的電腦網路運作以來，始終都有簡化使用者之資訊交換的需要。由於大型網路（如 Internet）和電腦通信的日益普及，因此，潛在的使用者人口也越來越多。雖然 XML 不是最通行的文件格式，但它具備其它文件交換格式沒有的優點。

4.XML 是最好的原始文件格式。

它能夠以應用程式（EDI，或電子資料交換）最適用的一或多種輸出格式（如 HTML、可攜性文件格式和 PostScript）來遞送內容。

5.XML 支援進階搜尋。

由於 XML 文件內容的結構和意義（由它的文法定義）是可以辨識的，因此在 XML 文件內，比較容易找到需要的內容。 XML 文件的一項需求是它們必須形式完整。「形式完整」的其中一個要求是每個起始標示都必須搭配一個結束標示。另一個需求則是，必須依照特定次序來建立標示的巢狀結構。這些需求簡化了文件的剖析和操作。

6.文法定義了 XML 文件的結構和內容，可用來進行更有效的搜尋。

比方說，使用者代理程式能支援依特定文法來編寫的文件集合之搜尋活動。另一個範例是搜尋曾提及由 "Ralph Higglebrewer" 所撰寫

之書籍。依文件內的標示名稱、標示屬性、資料內容和位置來搜尋，是使得 XML 文件更容易實作的其它搜尋策略。

7.XML 會區分文件結構、內容和呈現方式。

當必須使用程式來動態產生 Web 文件內容時，區分文件內容和文件結構便顯得格外重要。這種區分使 Web 團隊人員（網頁作者、商業邏輯程式設計師和圖形設計者）能夠平行運作，並降低彼此間的作業干擾。

8.XML 改進使用者回應、網路負荷和伺服器負荷。

XML 實作可讓 Web 伺服器同時傳送 XML 文件及其相關 XSL 樣式表給從屬站。每個樣式表都可提供所有或部份文件資料的不同檢視畫面。使用者可以選取適用的樣式表。變更檢視畫面（樣式表）時，不需要傳送另一個要求給伺服器。

9.XML 支援 Unicode。

XML 應用程式支援許多種文件編碼方式，包括 Unicode 雙位元組字集 (UTF-16) 和壓縮版本 (UTF-8) 。因此，XML 文件幾乎可以併入任何語言和 Script。 10.XML 支援進階的文件鏈結。 在 HTML 中，<A> 標示會將文件鏈結到另一份文件，或鏈結到同一份文件內的某個目標。這些鏈結都是單向的（從來源文件到目標文件）。<A> 標示包含目標鏈結的位址 (URL) 和鏈結的文字標籤。

2.2.3 XML Web Service 結構

XML 輕易地推翻我們建構及使用軟體的方式， 使用者與應用程式溝通的方式已因為 Web 而截然不同。藉由提供通用的資料格式，讓資料可以更輕易地被採用以及轉換，X ML 正逐步革新應用程式相互通訊以及電腦間互動的方法。以 XML 為基礎的標準，包括 SOAP 與 UDDI，構成了應用程式對應用程式通訊的開放方法，這就是所謂的 XML Web 服務。而 XML Web Service 能讓應用程式共用資料，且不論抓用何種程式語言，均可跨平台呼叫 XML Web 服務由圖 2.1 了解 XML Web 服務可以讓應用程式之間透過 Internet 進行通訊， 不論作業系統或程式

語言為何。XML W eb 服務可實作於任何平台，並且是透過公用標準組織 (例如 W3C) 來定義。使用 XML Web 服務時， 應用程式不但可以共用資料， 還可從其他應用程式呼叫功能， 不論該應用程式的建構方式為何。此外， 應用程式透過 XML 共用資料時各自獨立， 又能同時與彼此構成鬆散的合作群組來執行特定工作。網站的功能是將資料展示給使用者， 是伺服器與使用者之間溝通的橋樑。而 XML Web 服務則提供直接管道，讓不同應用程式間彼此互動。電腦內部的應用程式以及遠端系統， 可以使用 XML 及 SOAP 訊息，透過 Internet 進行通訊。Web Service 的優點:1.允許在不同平臺上、以不同語言編寫的各種程式以基於標準的通信。

2.使用標準的 Web 協定 - XML、HTTP 和 TCP/IP。許多公司都已經建立了 Web 基礎結構。

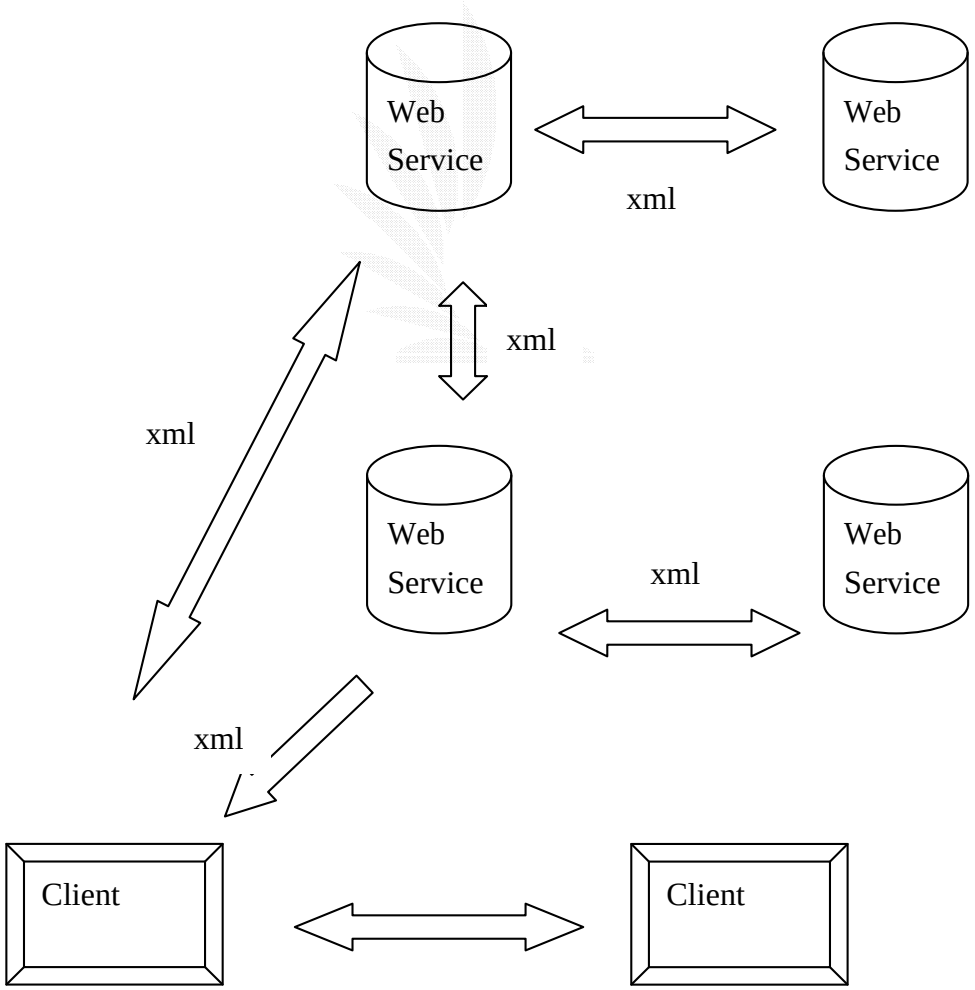


圖 2- 1XML Web Service 功能流程圖

2.2.4 DTD 使用

DTD(Document Type Definition)這項技術可以做到大家交換資料的標準， 如果大家採用共同的 DTD， 如此資訊交流便可以達到無障礙的空間。DTD 使用了正式的文法來詳細說明 XML 文件的結構與可允許的值。再者，DTD 定義出一份 XML 文件的辭彙， 這使得 XML 文件可以涵蓋每一個個案，而且所使用的詞彙都是定義於同一份 DTD。另一方面，在 XML 文件的驗証，可以透過 DTD 來驗証一份文件是否符合其架構與辭彙，如此一來，XML 可以充分的利用標示語言，且藉著 DTD 的定義，規定出 XML 文件的架構與辭彙，使得我們往後在使用 XML 文件，更為有利。

但是 DTD 還是有以下 4 項缺點:

(1) DTD 使用的是自己的語法，和文件實例不一樣。
(2) DTD 是封閉的架構，包含了 XML 文件的辭彙規則，卻不易延伸 DTD，片段宣告也無法整合關連
(3) DTD 在資料類型上也很短缺，無法定義其它類型。
(4) DTD 使得合乎語法的 XML 文件更嚴謹與精確，增加編輯 XML 文件時的困難度。

此外，DTD 對於 XML 應用程式有三項特色：

(1) 在商業領域模型具有正確且清晰的文件。
(2) 模型和 XML 剖析器之間以標準的方法來溝通。
(3) 驗証剖析器會找出 XML 文件的錯誤。

2.2.5 XML Schema 基礎

W3C 提供了一種稱為 XML Namespace (名稱空間）的規格來讓我們定義元素內容。XML N amespace 與 XML Schema 解決了一些使用 XML 上的問題。

(1) 對於複雜的問題的辭彙，提供了更好的組織性。
(2) 在轉換 XML 時，有更強的資料型別檢查。
(3) 較 DTD 更為精準明確，且富有彈性與延伸性。

(4) 不須要增加剖析器的複雜度，可以直接用 XML 的剖析器來存取辭彙定義。

以下針對 XML Schema 及 XML Namespace 作進一步探討。

XML Namespace 解決名稱含糊性與名稱衝突，根據 W3C 的推薦標準，名稱空間是：名稱的集合，並由一個 URI 參照來判斷，在 XML 文件中作為元素類型及屬性名稱使用（URI 是伺服器上 DTD 或綱要的位址）。XML Schema 是以 XML 的語法所撰寫。它允許多重名稱空間的使 用，並提供了強大的內容型別，此外，它也是 DTD 功能的超集合（superset），功能比 DCD（Document Content Description） 強， 但比 RDF（Resource Descriptor Frame work）遜色，，XML Schema 規格分成 Structure 和 Data Types 兩個部份。

Structures 是用來處理元素及屬性的描述與宣告，它允許 XML 設計者指定複雜的元素結構並設定內容允許值的限制。Datatype 則是用來設定內容資料型別的標準集合。 所以 XML Schema 與 Namespace 的結合，幫助我們將現有的綱要延伸，或將大型的複雜的綱要分割成小部分，也使得 XML Schema 的 重覆使用性增強，也解決了 DTD 各項的缺點。

以下列出 XML Schema 的效益：

(1) 透過辭彙片段的使用而具有較佳的組織及複雜工作的重複使用性。

(2) 轉換 XML 時具有優秀的資料型別。

(3) 保留 DTD 的精確性。

(4) 以 XML 的語法，這允許我們可使用傳統的 XML Parser 來處理 XML Schema。

2.3 密碼學概念

密碼學是研究加密和解密變換的一門科學。通常情況下， 人們將可懂的文本稱為明文； 將明文變換成的不可懂的文本稱為密文。把明文變換成密文的過程叫加密； 其逆過程， 即把密文變換成明文的過程叫解密。明文與密文的相互變換是可逆的變換， 並且只存在唯一的、無誤差的可逆變換。完成加密和解密的算法稱為密碼體制。在計算機上實現的數據加密算法， 其加密或解密變換是由一個密鑰來控制

的。密鑰是由使用密碼體制的用戶隨機選取的， 密鑰成為唯一能控制明文與密文之間變換的關鍵，它通常是一隨機字符串。

2.3.1 密碼學概念

2.3.1.1 密碼分析學：

是研究破譯密碼的一門科學。無論任何一種加密算法都是公開的。密碼學是用一種簡單的科學方法來保護資料， 因而所有人讀資料都是有經過授權的； 除了簡單的加密資料來保護資料， 密碼學的技術也被應用於：

＊確定被傳送的資料在接收之前是未改變過的 。
＊確認被傳送的資料確實是來自送件者。
一般有兩種類型密碼學被使用；symmetric key (對稱性的鑰匙) 和 public key (公開的鑰匙)(也叫非對稱的鑰匙) 密碼學。

2.3.1.2 對稱性密碼學：

對稱性密碼學也叫對稱密鑰加密， 它使用單把密鑰。這種密鑰既用於加密， 也用於解密。對稱密鑰加密是加密大量數據的一種行之有效的方法。對稱密鑰加密有許多種算法， 但所有這些算法都有一個共同的目的- 以可還原的方式將明文（未加密的數據）轉換為暗文。暗文使用加密密鑰編碼， 對於沒有解密密鑰的任何人來說它都是沒有意義的。由於對稱密鑰加密在加密和解密時使用相同的密鑰， 所以這種加密過程的安全性取決於是否能保證機密密鑰的安全。 舉一個簡單的對稱性鑰匙密碼學的範例，假設從朋友處收到一個通知， 你和你的朋友同意來加解密你們的訊息，你們將使用下列演算法； 每個字母將會上移三個字母， 例如 A=C， B=D， 而 Y 和 Z 轉一圈回到 A 和 B。這個方程式 ("每個字母上移三個字母") 就是送信者使用來加密訊息的鑰匙； 而收信者使用相同的鑰匙來解密，任何人如果沒有鑰匙就不能夠讀此訊息。因為相同的鑰匙視同實用來加密及解密訊息， 這個方法是一個對稱鑰匙的演算法。這類的密碼學即是我們所知的秘密鑰匙密碼學，因為此鑰匙 必須被秘密保存於送信者和收信者，以保護資料的

完整性。

2.3.1.3 非對稱性密碼學

公鑰加密使用兩個密鑰- 一個公鑰 和一個私鑰，這兩個密鑰在數學上是相關的。為了與對稱密鑰加密相對照，公鑰加密有時也叫做不對稱密鑰加密。在公鑰加密中，公鑰可在通信雙方之間公開傳遞，或在公用儲備庫中發佈，但相關的私鑰是保密的。只有使用私鑰才能解密用公鑰加密的數據。使用私鑰加密的數據只能用公鑰解密。由於它用私鑰加密的數據只有公鑰才能還原， 這被用在數位簽章中。每支鑰匙都具有改變內文的功能，私有的鑰匙 產生一個私有改變內文的功能，而公開的鑰匙 產生一個公開改變內文的功能，這些功能是反向相關的。例如，如果一個功能是用來加密訊息，另外一個功能則被用來解密訊息，不論此改變內文功能的次序為何皆不重要。公開的鑰匙系統的優勢是兩個使用者能夠安全的溝通而不需交換秘密鑰匙。例如，假設一個送信者需要傳送一個信息給一個收信者，而信息的秘密性是必要的，送信者以收信者的公開的鑰匙來加密，而僅有收信者的私有的鑰匙能夠對此信息解密。公開的鑰匙密碼學是非常適合於提供認證完整和不能否認的服務，且公鑰密碼中的私鑰無法記憶，用對稱密碼加密公鑰密碼中的私鑰是不錯的用處。

2.4 Des 加密演算法介紹

Des 演算法是一種數據加密演算法，從 1977 年公佈以來，一值是國際上的商用保密通信和電腦通信的最常用的加密標準。DES(Data Encryption Standard)即數據加密算法，是 IBM 公司於 1977 年研究成功並公開發表。應用方向如信用卡持卡人的 PIN 的加密傳輸、IC 卡與 POS 的互相認證、金融交易中的密碼鍵盤等，均用到 DES 演算法

2.4.1 Des 演算法原理說明

- 1970 年中期由美國 IBM 公司發展出來，屬於區塊加密法，所有區塊大小控制於 64bits，進行加/ 解密，分為母金鑰作為 DES 加/ 解

密的金鑰，子金鑰須按順序排列加解/ 密，由 DES 針對母金鑰產生。

- 所使用的金鑰 64 bits 有 8 bits 作為除錯用，真正金鑰只有 56 bits 稱為母金鑰，子金鑰由 56 bits 母金鑰輸入運算法中所產生的一系列密鑰。
- DES 的安全性可由下列 6 個方面討論：
  - 3.1.弱金鑰：由於子金鑰產生過程設計不當所導致，產生出少數金鑰降低 DES 安全性。使得原設計 Dk=(Ek(m))=m 或 Ek=(Dk(m))=m，其中 m為明文 Dk 為解密、Ek 為加密。弱金鑰將使另一運算式也成立 Dk=(Dk(m))=m 或 Ek=(Ek(m))=m，將大大降低 DES 安全度。
  - 3.2.半弱金鑰：子金鑰產生系統滿足下列 2 種條件。
    - 3.2.1.子金鑰排程中的左旋(或右旋) 運算暫存器只含有”¨010”，”¨010”，…K”¡¨01”或”¨101”，”¨101”，…K”¡¨10”兩種。
    - 3.2.2 另一左旋(或右旋) 運算暫存器只含有”¨000”，…，”¨111”，…，”¨010”，…，”¨101”，…任何一種形式的資料。
  - 3.3.DES 的互補性(Complement)：DES 中明文 m、密文 C、金鑰 K之間存著互補的特性。簡單由下列式子表示：
  - 3.3.1.若 Ek(m)=C 則 Ek’¦(’¡¦)’，則破解者如何取得 Ek(m) 與 Ek’¦(’ 即可由此找出破解的漏洞，但機率卻是相當的小。
  - 3.4.替換盒的設計原則，將維繫整個 DES 加密系統的安全性，它的設計過程也一直因為安全考量而所之有限，1977 年美國國家安全局曾針對此議題於第 2 次 DES 會議中提出替換盒設計方案。
  - 3.5.DES 的 16 個回合：Alan Konheim 提出超過 8 個回合的 DES，其密文與明文合金鑰的關係已經是隨機關係了。
  - 3.6. 差分攻擊法：基本上是屬選擇明文攻擊法，就是分析特殊明文配對的差質對於其所相對應的密文配對之差質(進行 XOR 運算) 所產生的影響，有些特殊的差值有很高的機率會再出現於密文配對上，我們稱此為特徵值(Character)，選出許多擁有此特徵的配對，進行加密後得到相對應的密文配對，再藉由密文與特徵值推導出可能的金鑰。

2.5 使用 java 實作加解密

2.5.1 密碼學簡介 － 加密與解密

加密是一個將欲加密的資料用一些數學運算轉成一團令人看不懂的東西的過程；解密則是將加密文轉換回原始文字的過程。這個過程中，扮演原始文字與加密文字之間轉換的數學演算法稱為 Cipher。

現代的 Cipher 多半會用 Key 來加密與解密資料。所謂 Key 是指一個機密值，我們可將它視為一通行密碼。加密文字必需使用對映的 Key 才能解密為原始文字。

對稱型 Cipher 於 javal：

對稱型 Cipher 在傳送端與接收端所用的 Key 是一樣的，如下圖所示，對稱型 Cipher 又叫 Private Key Cipher，因為 Key 的值只有傳送端和接收端知道。如果有第三者知道了 Private Key 值，也就能解開加密的資料。

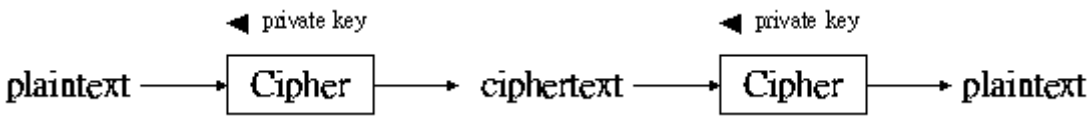


圖 2-2 對稱型 Cipher 的運作

2.5.2. JCE

2.5.2.1 JCE 下載

因為美國法規的限制，Sun 在 JDK 裡只提供了少數的加密方法，其餘大部份則只在 SunJCE 裡提供，而且 SunJCE 的 API 限制只有美國、加拿大地區可以下載。表 2-1 為 Sun 及 SunJCE 分別支援的加密演算法。

	名稱	型別
Sun	MD5	訊息摘要
	SHA-1	訊息摘要
	DSA	簽章
SunJCE	HmacMD5	MAC
	HmacSHA1	MAC
	DES	對稱型 Cipher
	DESede	非對稱型 Cipher
	PBEWithMD5AndDES	對稱型 Cipher
	DH	Key 的交換

表 2-1Sun 及 SunJCE 支援的加密演算法

雖然美國法規有這樣的限定，但是在美國境外也已經有廠商實作出 JCE，並且可以在網路上直接下載，表 2-2 就是下載網址的列表。

套件	網址	免費
JCE	http://java.sun.com/products/jdk/1.2/jce/	是
Cryptix	http://www.cryptix.org/	是
IAIK	http://wwwjce.iaik.tu-graz.ac.at/	否

表 2-2JCE 軟體下載網址

2.5.2.2 JCE 安裝

使用 **Java 1.4.1(**函以上版本**)**

- 到以下兩處資料夾找到 java.security 檔
  - C:\jdk1.4.1\jre\lib\security
  - C:\Program Files\Java\JRE\1.4.1\lib\security

  Ps：C:\jdk1.4.1\ 及 C:\Program Files\Java\JRE\1.4.1\ 為安裝 java 的參考路徑

- 在將 java.security 檔案用筆記本或文書處理器開啟，找到裡面的兩行字串

```
security.provider.1=sun.security.provider.Sun       //原有
security.provider.2=com.sun.rsajca.Provider         //原有
```

- 插入

  “security.provider.3=com.sun.crypto.provider.SunJCE”

  此行字串如下：

```
security.provider.1=sun.security.provider.Sun       //原有
security.provider.3=com.sun.crypto.provider.SunJCE  //新增
security.provider.2=com.sun.rsajca.Provider         //原有
```

- 兩個資料夾的 java.security 檔都要修改才能正確運作

使用 Java 1.3.1

- 安裝 jre 到以下網址下載 jre 並安裝完成

http://java.sun.com/j2se/1.3/download.html

- 下載 Java Cryptography Extension (JCE) 1.2.2

  ◆ 方法一：到 sun 網站下載

由於當初加解密軟體被美國國防部列為管制項目，雖然現在漸漸開

放，但是還是受到限制，因此要下載需要先通過註冊程序
http://jsecom15c.sun.com/ECom/EComActionServlet?StoreId=22&PartDetailId=JCE-1_2_2-G JS&TransactionId=Try&LMLoadBalanced=

- ◆ 方法二：到以下硬碟空間下載(暫時)
  http://140.134.4.20/~d8956836/jce-1_2_2.zip

- 將 jce 解壓縮後，到解壓縮目錄夾的 jce1.2.2/lib/ 資料夾中取出以下四個 jar 檔
  - jce1_2_2.jar
  - local_policy.jar
  - sunjce_provider.jar
  - US_export_policy.jar

- 將以上四個 jar 檔複製到
  - C:\jdk1.3.1_09\jre\lib\ext
  - C:\Program Files\JavaSoft\JRE\1.3.1_09\lib\ext

  Ps：C:\jdk1.3.1_09\ 及 C:\Program Files\JavaSoft\JRE\1.3.1_09\ 為安裝 java 的參考路徑

- 其餘步驟同使用 Java 1.4.1

2.5.3 使用 java 實作 des 演算法加密與解密步驟

這裡舉的加密/解密是屬對稱型 Cipher; 在金融交易裡，常用對稱型 Cipher 來加/解密資料。

加密/解密的步驟:

- 利用密碼產生一把 key
- 呼叫 getInstance 產生一個 Cipher 物件
- 呼叫 init 設定為加密或解密
- 加密/解密

第三章系統分析與架構

3.1 系統規劃

3.1.1 瞭解問題

在於改善網路的傳輸的安全問題,可以大量改善傳輸安全的顧慮,增加網路安全的傳輸。對於資訊爆發的時代中,傳輸的格式和應用能力不斷提升,借由系統的擴充性,可不斷的增加原本的功能,使系統不被淘汰,可以依據新功能的增加,讓效能更加強大,並可以提供私人企業的傳輸處理,讓功能有專屬性。下面有一個功能需求表,可以很輕易的去看出現在的需求和系統的改變。

目前的網路傳輸	系統的網路傳輸	效能
網路傳輸檔案,如果遺失就必須重新傳送	擁有續傳的功能,	可以使傳輸的檔案,依據失去檔案,而去重傳,這樣可以節省許多的時間
擁有的傳輸功能特定,沒辦法增加功能套件, 沒有太大的變化	程式是由物件導向的方向去撰寫,可以方便的去搭配額外的套件程式	讓能的擴充性十足,以方便日後的發展迅速,避免被淘汰的問題
網路傳輸時,如果需要用套件(壓縮..等)需要額外的去使用	基本的一些套件,可以內建在系統中,方便使用	可以方便的去使用許多的功能,而且可以依據你要的功能選擇,有多樣性的功能選擇

表 3-1 建立功能需求表

3.1.2 確認效益

確認增加系統在後功能的確認效益以下幾點：

1. 利用資料輸入可以直些選取要的傳輸功能(如: 加密, 壓縮)，減少使輔助套件的時間。
2. 使用續傳資料傳輸降低遺失率，提高資料傳輸的可靠度。
3. XML 的轉換讓傳輸可以擁有 XML 的優點, 提高資料的提升性和便利性。
4. 可以銜接任何套件一起使用(如: 掃毒)，使系統的衝性大大增加。
5. 程式的跨平台特性, 可用於各種系統。

這些的功能, 可以徹底的改善傳統的網路傳輸處理, 提高工作的效能和方便性, 不僅是一般的功能都有, 卻更涵蓋了現在的新潮工具, 而目標是讓網路的傳輸性和安全性提高到最好的狀態, 讓使用者能 100% 的使用網路傳輸。

3.1.3 估計時間與成本

此系統的專案是由三個人去開發, 而開始的資料收集、程式撰寫、系統的選擇使用, 會花費比較多的時間, 而系統的資料由書本、網站取得相關資料。而續傳等附加觀念, 一起開發討論。專案時設定為一年，採漸進式模組發展，由一開的的資料建立模組, 分析所需要的系統, 建構之後, 開始選定系統及開發軟體, 慢慢開發模組, 而模組的架構也是漸進式的, 每一個模組有特定功能和其他模組是互相依賴的。開發環境所需成本沒有底限, 均是使用免費的開發軟體及試用版的工具(如下圖)。如果企業要使用系統的資源, 能然需要徵求軟體商的同意, 而開放的程式就可以自行取用。

系統配備	成本
電腦一台	不限
網路費	不限
JAVA 套件	免費
XML 套件	免費

表 3-2 系統成本表

3.1.4 確認專案範圍與限制

本系統的專案範圍定為為一般的私人企業傳輸,而鎖定的目標是讓了各企業間,擁有它們特定的網路傳輸路線,而不是共用性的去傳輸，讓兩個企業能隨時的傳送機密資料。使用者的定位是主要的 IT 人員、系統的操作人員。

專案開發及實作方面,主要有兩台電腦,而每一台主機都擁有它的單獨 IP,就可以去實作系統並且去開發測試。而電腦的作業系統主要是任何平台系統都可以使用。

3.2 需求報告書

3.2.1 需求輸入

1.檔案切割資料: 傳送資料時, 資料切割大小資料
2.掃描時間資料: 每隔多少時間就會掃描目錄的資料
3.緩衝區資料: 程式運作的緩衝區大小資料
4.接收端傳送資料: 接收時的 port 設定資料
5.傳送端接收資料: 傳送時的 port 設定資料

6.錯誤時嘗試資料: 當傳送有錯誤時所要嘗試次數資料
7.停止時間資料: 當傳送錯誤欲在嘗試之前停止時間資料
8.掃目錄路徑資料: 掃描目錄的路徑資料
9.接收目錄路徑資料: 接收檔案欲放的目錄路徑資料
10.是否分割資料: 決定是否分割的資料
11.是否加密資料: 決定是否加密的資料
12.是否壓縮資料: 決定是否壓縮的資料
13.暫存檔目錄路徑資料: 暫存檔目錄的路徑暫存檔目錄

3.2.2 需求輸出

1.設定檔: 基本要傳輸的規格設定用來決定傳輸的功能。
2. XML 檔案: 傳輸端經過處理後的 XML 檔案。

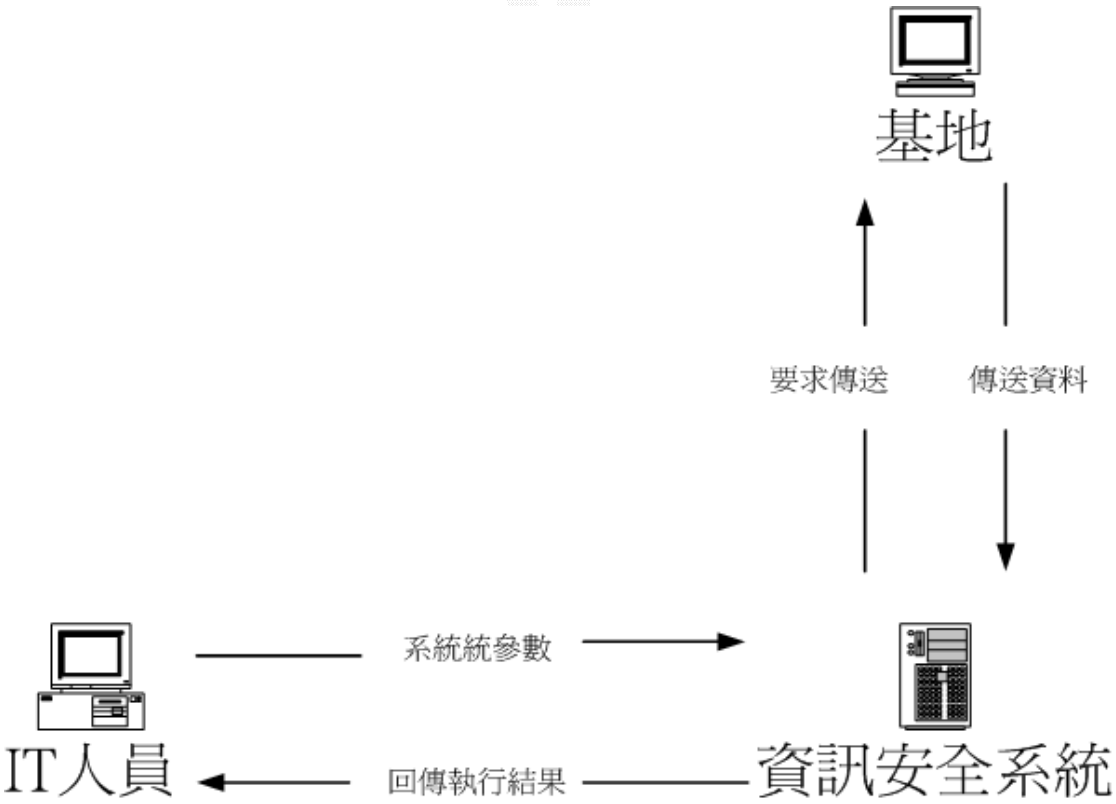


圖 3-1 系統背景圖

3.2.3 過程

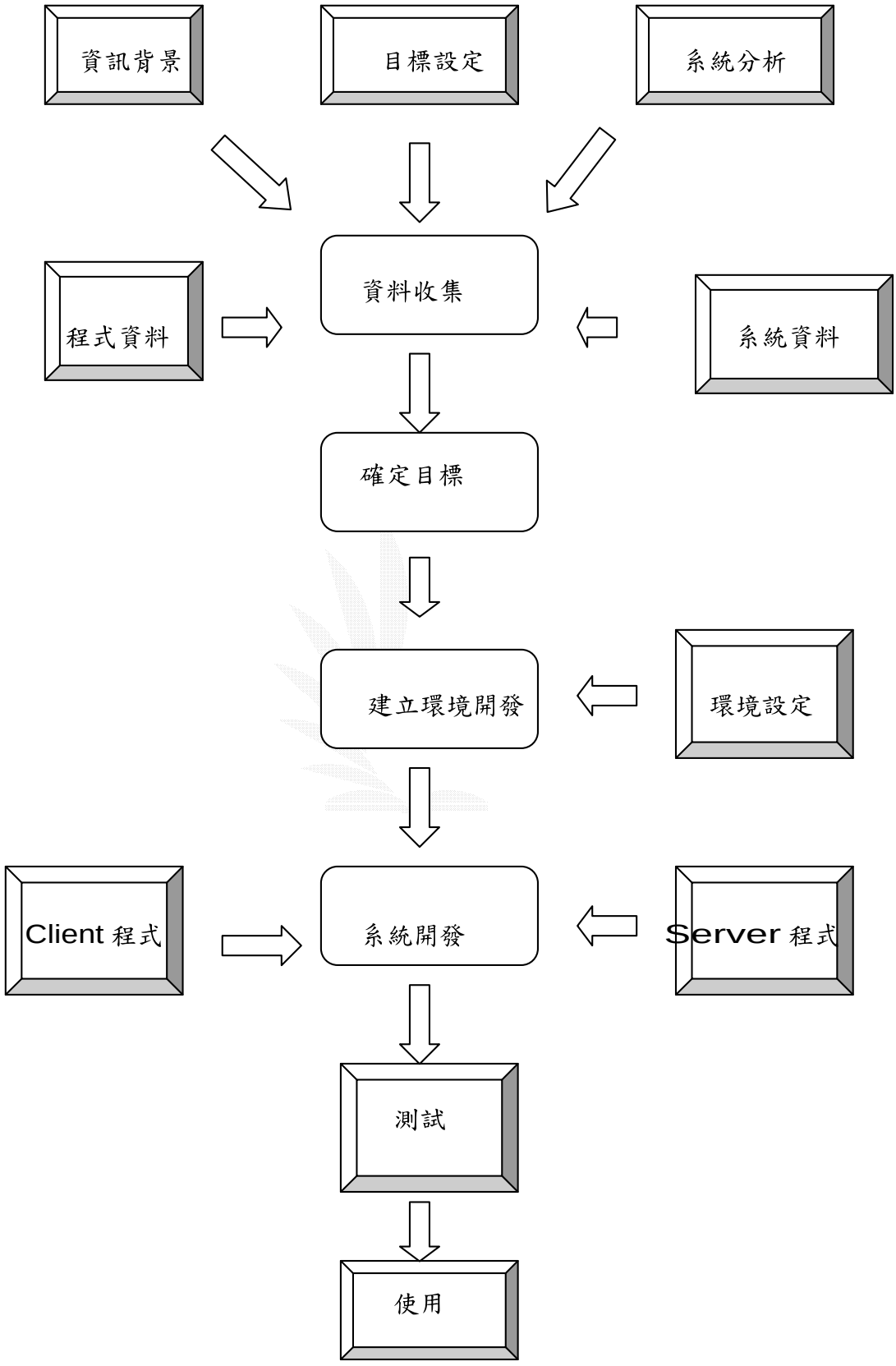


圖 3-2 策劃規劃流程圖

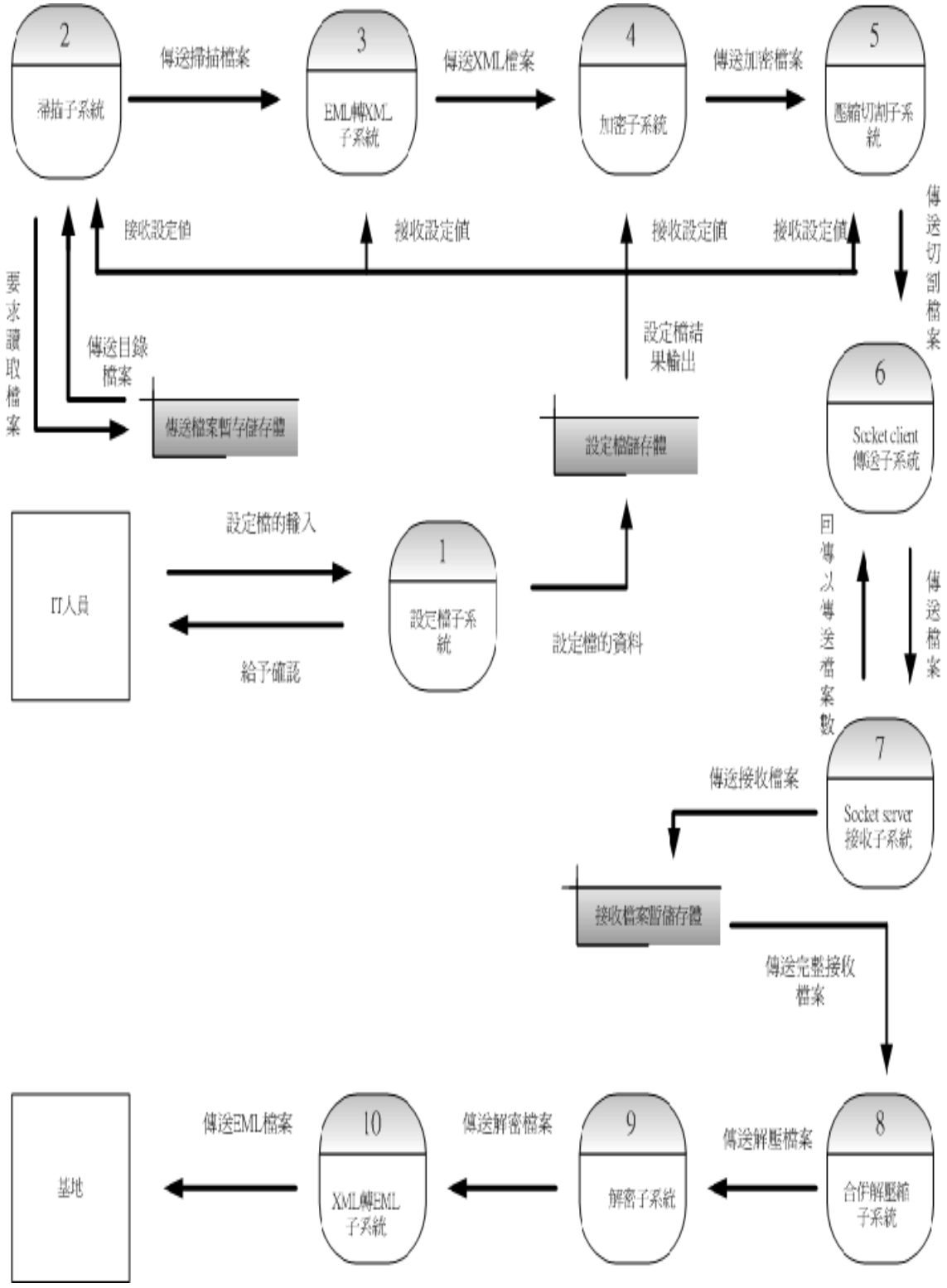

2
掃描子系統
傳送掃描檔案
3
EML轉XML
子系統
傳送XML檔案
4
加密子系統
傳送加密檔案
5
壓縮切割子系統
傳送切割檔案
接收設定值
接收設定值
接收設定值
接收設定值
要求讀取檔案
傳送目錄檔案
傳送檔案暫存儲存體
設定檔結果輸出
設定檔儲存體
6
Socket client
傳送子系統
IT人員
設定檔的輸入
給予確認
1
設定檔子系統
設定檔的資料
回傳以傳送檔案數
傳送檔案
7
Socket server
接收子系統
傳送接收檔案
接收檔案暫儲存體
傳送完整接收檔案
基地
傳送EML檔案
10
XML轉EML
子系統
傳送解密檔案
9
解密子系統
傳送解壓檔案
8
合併解壓縮子系統

圖 3-3 系統 DFD 圖

3.2.4 時間安排

1. 掃描目錄的間隔時間可以自行設定,設定後會不斷的執行。
2. 錯誤傳送的時間可以自行設定。
3. 設定檔是使用傳送時,至少必須設定一次。
4. 暫存檔的刪除，固定傳送完就執行一次。

3.2.5 控制

1.操作人員: 直接於系統中操作系統。
2.IT 人員: 統主要作為系統維護,程式更新,系統參數更新之用。

3.3 系統設計

3.3.1 輸入

3.3.1.1 email 格式

我們所設計出來的系統就是要能自動的抓取欄位的值，並分析判斷他的用途後，對應到我們所需要的資訊中。因此，只要使用者編輯完信件之後，在已定的欄位中填入所需的值，送到傳信件的資料夾中即可完成多功能傳信的手續。

email 有一些重要的欄位及說明如下表：

"From:"	送信者的 email 位址
"To:"	收信者的 email 位址
"Cc:"	副本抄送
"Bcc:"	密件副本
"Subject:"	主題
"Date:"	日期
"Size"	檔案大小
"Name"	名稱
"Copy"	備份
"Pop3mail"	是否為 pop3 的 email
"Trashcan"	刪除
"Peruse"	是否閱讀過信件
"Message-ID"	信件編號
"compress"	0-９０:不壓縮 1~9：壓縮率
"encrypt"	0-４０：不加密 1-4：加密模式
"MIME-Version"	MIME 格式版本
"Body"	信件內文（MIME 內文，除 attach 之外的）
"Content-Type"	以下即為附加檔案

表 3-3email 欄位表

3.3.1.2 介面設計

即使是所需要的重要資訊都在 email 之中，但是有些參數是整個系統固定的，他不適合放在 email 的欄位中，如 server 的 ip 位址。諸如這些設定值，我們寫了一個設定檔 system.conf 來紀錄他，故程式一執行即會開始讀此檔運作。但在不同的電腦會有不同的環境，因此在執行之前勢必還要做必要的修改設定，而使用者介面就可以解決這個問題。

在我們原本規劃的系統中，使用者介面並不存在，因為所設計的系

統是一套一執行就全自動運行的系統，他不需要使用者做控制的動作，因為執行開始之後一切的輸入直接是由 email 內的 tag 欄位的值決定，設定值也由系統的設定檔設定好，因此正確，穩定且有效率的運行是我們最主要的設計方向。

但是，在經過系統分析之後發現，若沒有使用者介面來搭配程式的運作，不但使用者操作不便，需要輸入多道指令才能完全啟動，就而對我們程式開發者而言，每次程式要在不同的電腦上運作前，就要去系統的設定檔中改資料，也的還需要改程式碼，然後再重新 compiler 過，使用介面前和使用介面後的比較表如下表。因此，現在又有了 GUI 的呈現。

	有圖形介面	無圖形介面
系統參數更改	可在 textfield 修改	需要改變程式碼，重新 compiler
變更掃描資料夾	可在 textfield 修改	變更程式碼再 compiler 或建立跟開發環境一樣的資料夾
安全性	有使用者登錄畫面	再命令提示字元下 java *** 的後面再加一串密碼
缺點	多等待了產生圖形的時間	操作不易

表 3-4 使用介面前和使用介面後的比較表

介面設計是以使用者為中心的設計，針對了程式在每台電腦不依樣的環境，做有彈性的調整，直到全部調整好了，才開始運作。

狀態視窗	來表達現在處理的狀態
** 輸入欄位*	來改變控制參數
說明標籤	說明欄位用途
控制按鈕	一切就緒後按下按鈕才開始執行

表 3-5 介面應要包含的元件列表

輸入欄位格式說明如下：

以下為放置在介面上 textField 的設定項目：

- buffer_size = 8192

buffer_size 是程式執行時每個物件所分配給予的 Buffer 緩衝區大小，用於檔案的處理，也就是當一個程序(例如：壓縮、加密)在執行的時候，都會配給這個項目設定的大小。單位：Byte (1/1000 Mb)

- send_port = 2222

傳送端主機所用傳送 Port Number。此項若沒有設定，則 Client 傳送程式會自行隨機找一個沒有正在服務中的 Port 使用。

- remote_port = 2223

接收端主機所用接收 Port Number。此項一定要設定，Client 端無法自行決定接收端使用的 Port Number，如果沒有設定，則程式無法正常運作。

- delay_time = 10000

傳送時，如果遇到傳送失敗，間隔一段時間再重新嘗試傳送
時間單位：ms (1/1000 Sec)

- try_times = 10

傳送時，如果傳送失敗，做到此項目設定的最多重新嘗試傳送次數

- divide_size = 5000

欲傳送的檔案分割大小，傳送端程式會依據這個設定，決定 divide_size 為每次要傳送多大。單位：Byte (1/1000 Mb)

- scan_time = 10000

傳送端程式會依據這個數據，每隔 scan_time 時間去掃描程式 readinput.java 中 scan_dir 所設定的目錄，也就是每隔多少時間掃描 *.eml 一次，若讀到一封 E-mail 則暫停掃描。

以下為 Server 端和 Client 端執行時所需要的目錄

◆ xml_dir

Client 端由*.eml 檔轉為*.xml 後，放置於這個目錄，等待後續的處理

(如：加密、傳送)

◆ temp_dir

為 client 執行中，放置暫存檔的位置，加密、壓縮必須用到

◆ recieve

Server 端接收到傳送端 Gateway 傳送的分段資料後，先放置於此目錄，並在這個目錄進行組合成，還原為原來的檔案。

◆ zipout

Server 端的暫存資料夾，和上面 temp_dir 不同，這個資料夾為接收端處理過程需要的資料夾，解壓縮或解密會用到。

◆ server_xml

Server 端將處理完成後成為 XML 檔的郵件放置覦此目錄，等待 Server 端程式將裡面的 XML 檔，還原為 *.eml 檔後，放至於前面所述 readinput.java 內的設定項目 server_out ，將檔案的控制權讓給其他的程式將 *.eml 讀出

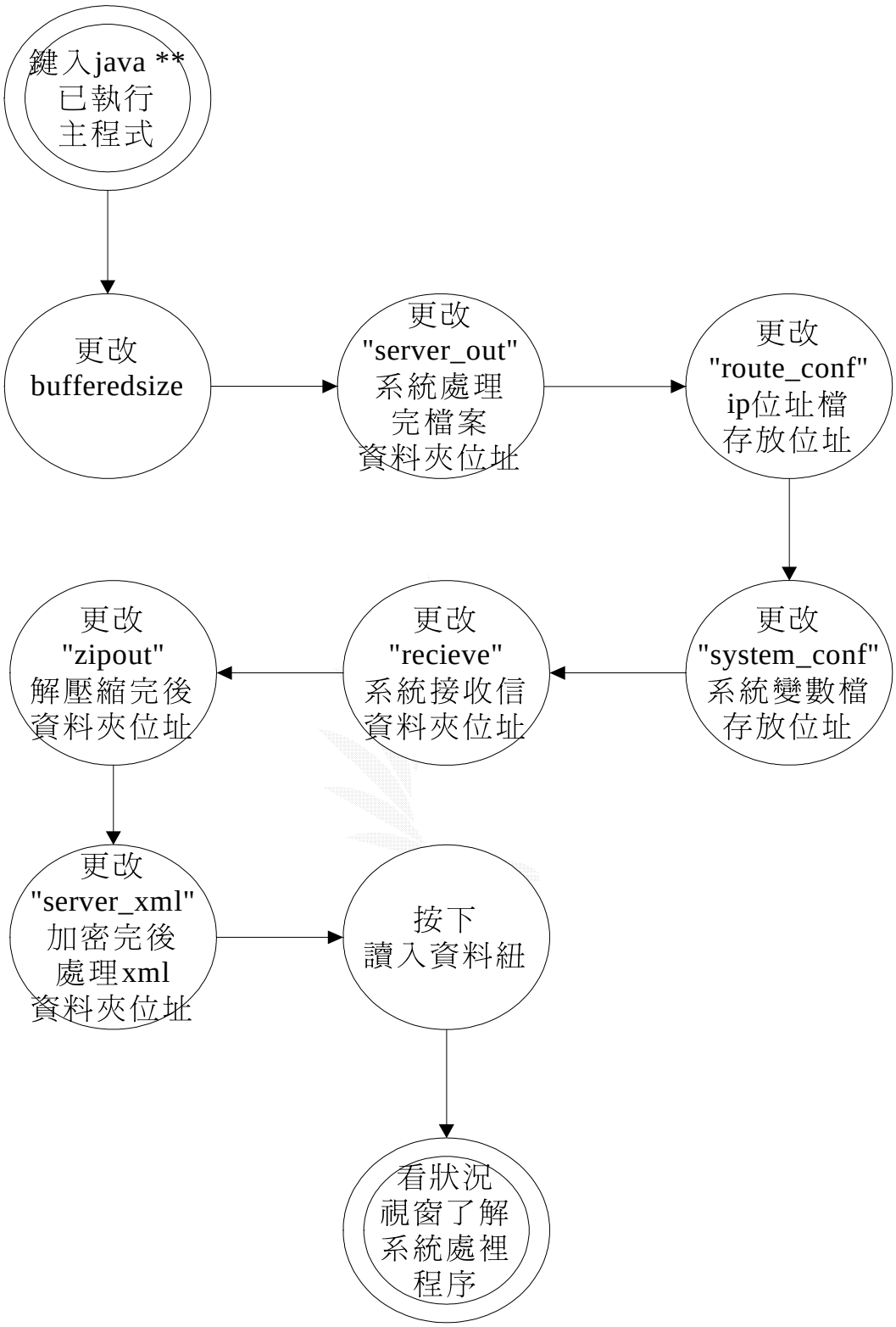


圖 3-4Server 端系統介面流程圖

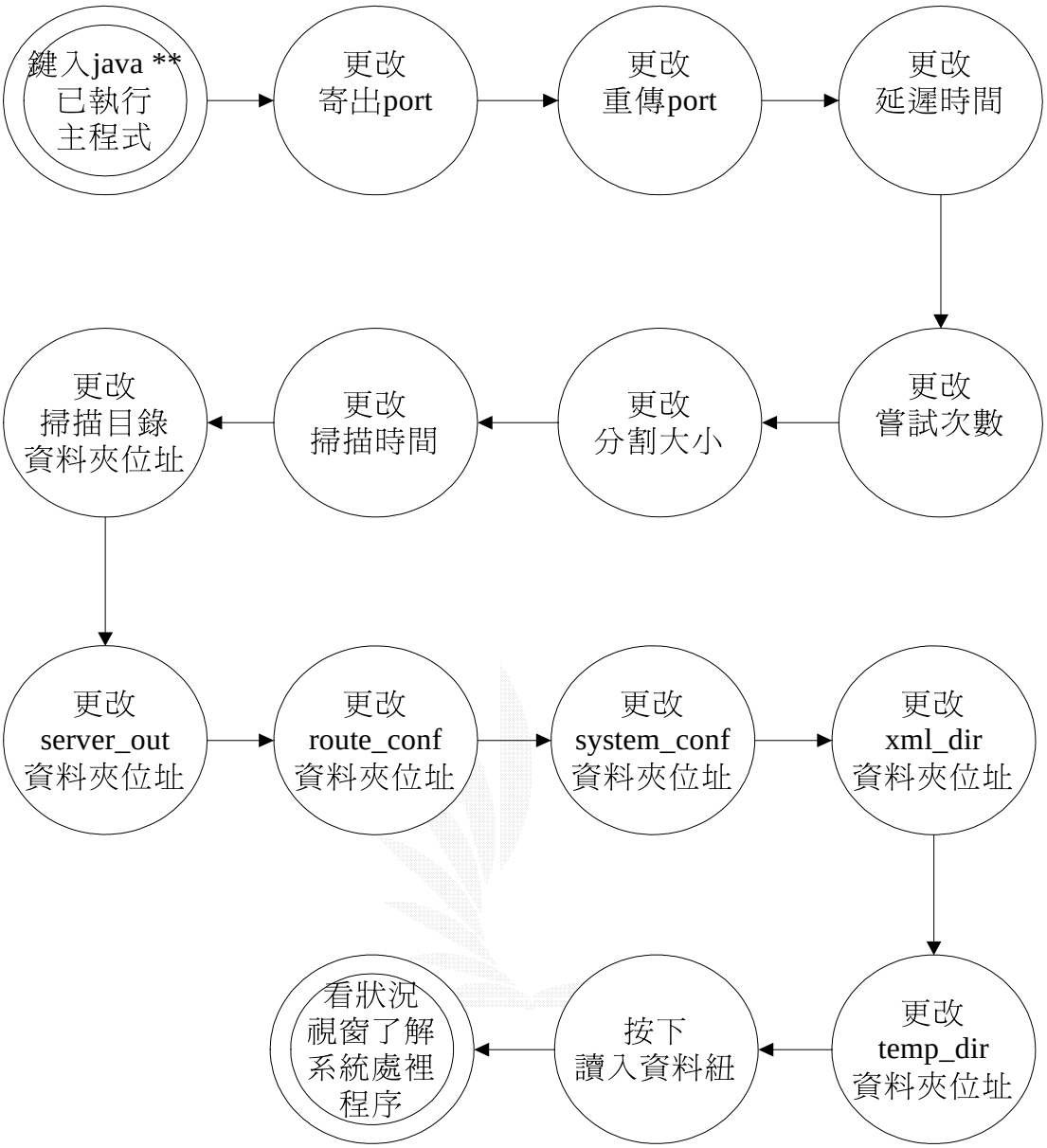


圖 3-5Clinet 端系統介面流程圖

3.3.1.3 輸入錯誤

如果輸入的值有錯誤，系統會依不同欄位，做出不一樣的處理步驟，有些沒有影響，照常執行，有些則會引發例外處理使此信發送不出。

3.3.2 輸出

輸出原本設計是將結果和狀態印在命令提示字元下，不過系統分析的時候已經將輸出的狀態視窗考慮進去。

3.3.3 Client / Server 設計風格

在軟體工程討論的，相對於分散式系統的就是 client / server 架構。而在用戶端/ 伺服端架構中又有分

- 胖用戶端(fat client)：程式的呈現層和應用處理層是在用戶端電腦上執行；遠端伺服器上只進行資料管理。 優點是有很好的用戶端效能，但是會使系統管理工作 (如，更新軟體) 變複雜。
- 瘦用戶端(thin client)：伺服器執行應用處理層及資料管理層的動作，而用戶端則只執行資料的呈現動作。

按照此分類，**我們的系統是** fat client model，以下是 server 和 client 的工作圖

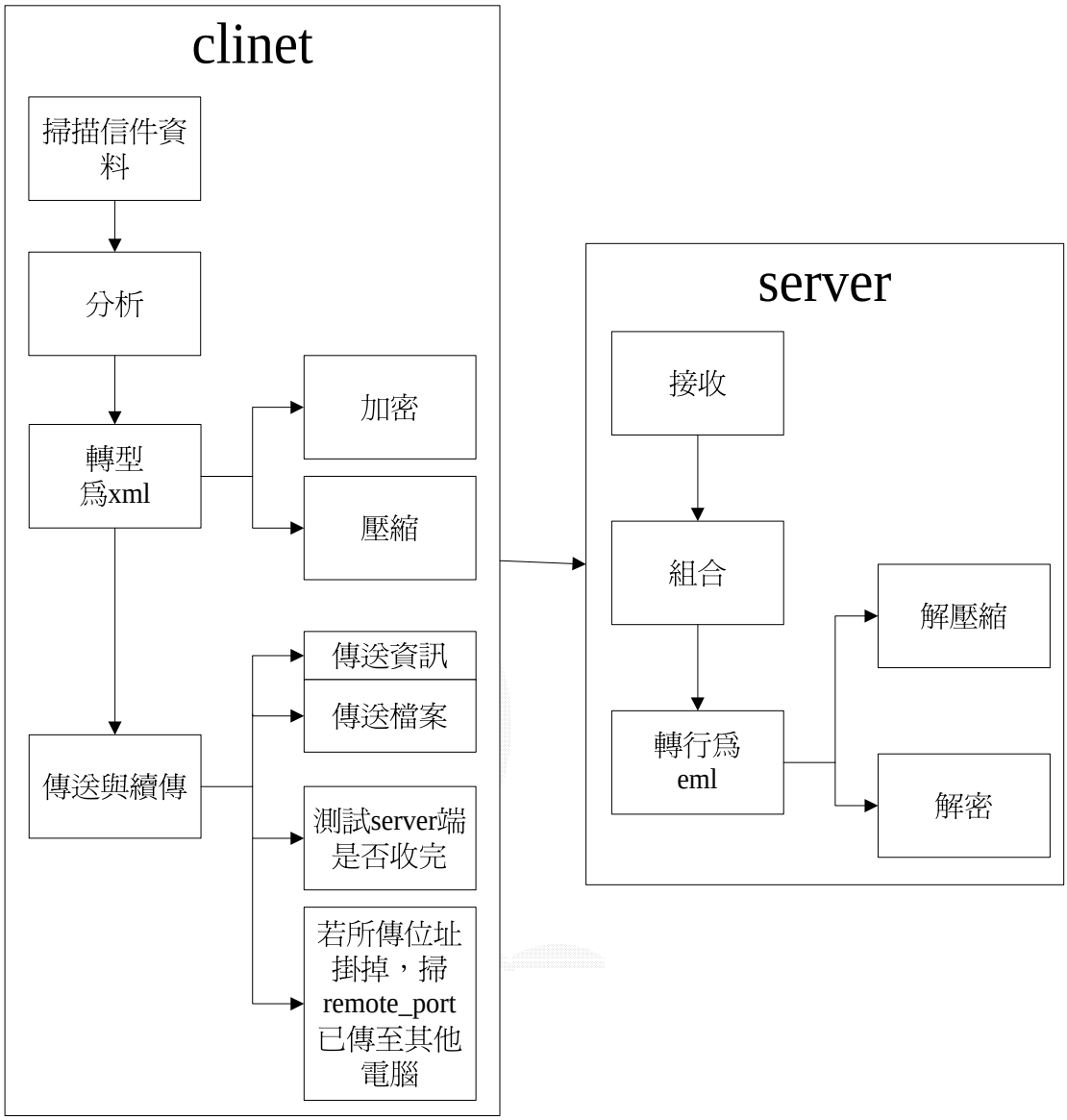


圖 3-6server 和 client 的工作圖

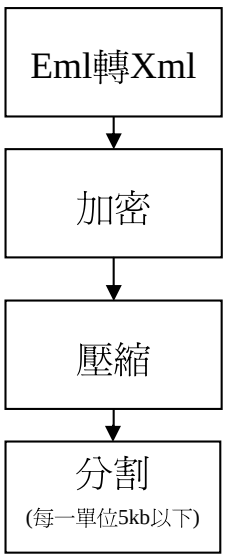
3.3.4 系統安全

雖然我們所寫的程式並不會對系統造成太大威脅，但是若是在公用電腦隨意的被不當使用，也會造成對作業系統效能的拖累，重要的是如果不慎會將不想發送的資料發送出去，當發送完成就會做砍檔動作，因此會導致資料流失。故此系統進入主畫面前還有需要輸入正確的使用者、密碼已達保護。

3.4 系統建置

3.4.1.演進式(Evolutionary prototyping)的設計方式：

當我們決定好系統的初步規劃之後，我們開始著手設計程式。設計的方式是先將小元件寫好，當每個功能元件做好之後，都會讓其 class 檔中有 main，以測試是否能運行或結果是否正確。當所要的元件設計的差不多了，才開使設計一個主控制各個元件的 main 函式，當 main 函式開發完成，我們做了第一次的測試。第一次的初版只有最完整功能的部分。最初版的功能如下圖



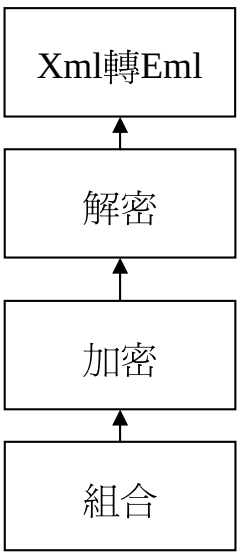
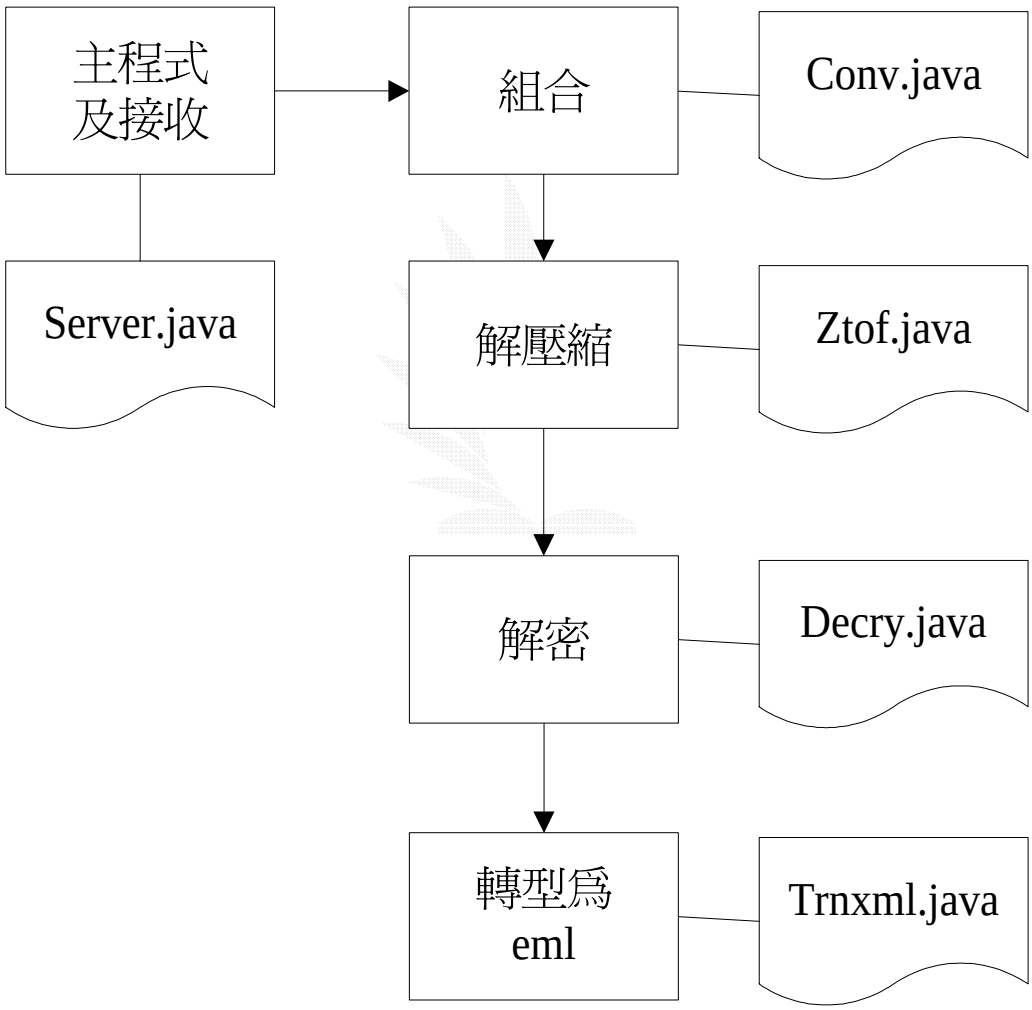


圖 3-7 client 初版模型　　server 初版模型

最初始的模型，還沒有把自動掃描目錄，掃描 email 欄位加進去，因此，在執行的時候要把檔案名稱加入到執行命令之中，而也沒有判斷是否要加密壓縮就直接處理了。當結果正確，我們才將我們的系統逐步完整化。

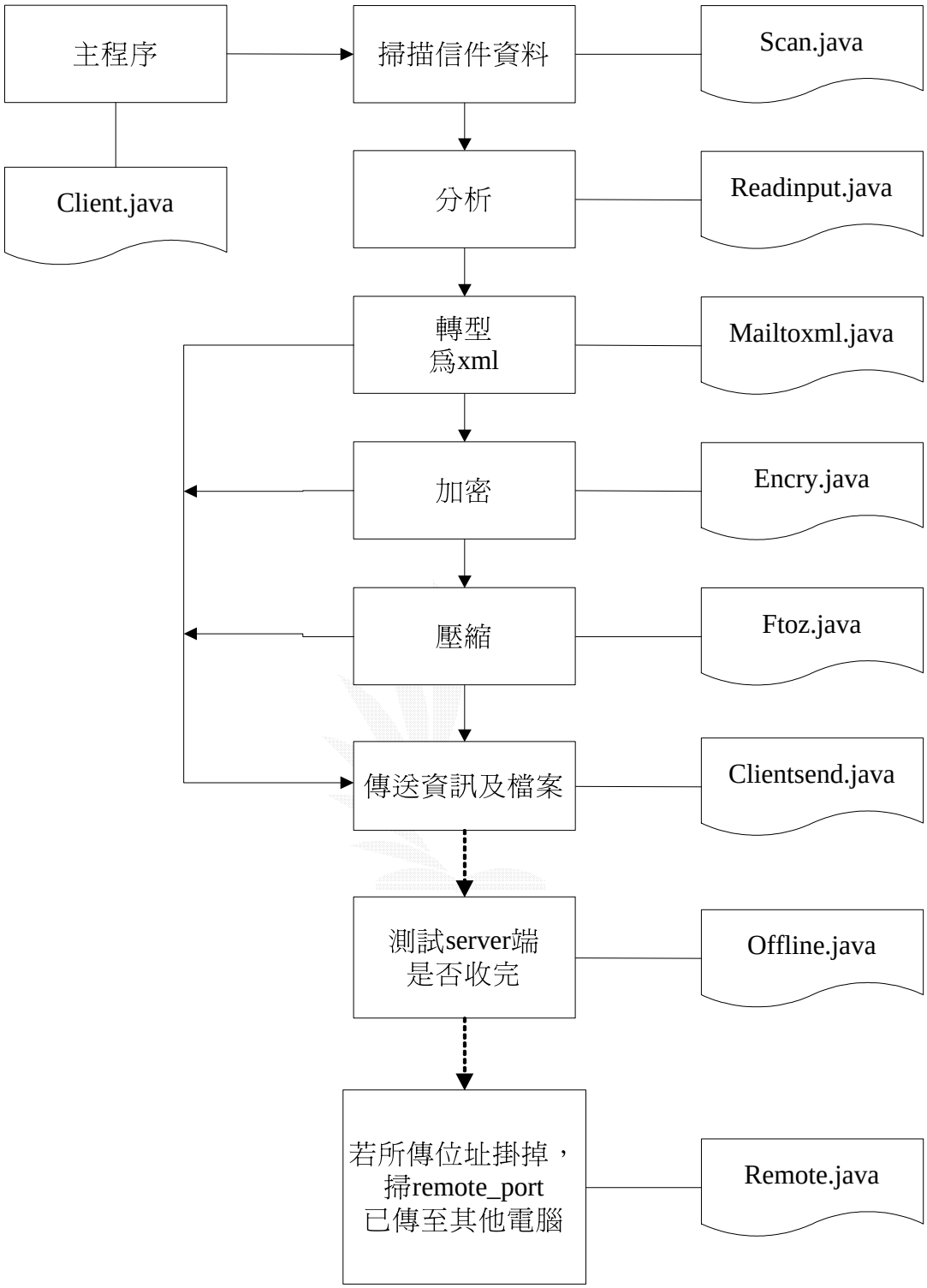
3.4.1.1 程式設計架構流程圖

以下為 server 端及 clinet 端架構的流程圖



ps:server.java 加了介面後為 serverwin.java

圖 3-8server 端設計架構流程(及對應所屬 java 檔)



ps:clinet.java 加了介面後為 clinetwin.java

圖 3-9client 端 設計架構流程(及對應所屬 java 檔)

3.4.1.2 程式設計 java 檔和包含的 class 檔一覽表

java 檔 (共 15 個)	class 檔(共 24 個)
Client 端 (共 9 個 java；16 個 class)	
readinput	conf
	reader
	analysis
clientwin	clientwin$1
	clientwin$Button1Mouse1.class
scan	filescan
	remind
	nameFilter
mailtoxml	eml_reader
	emltoxml
	addfile
remote	sendto
encry	encry
ftoz	ftoz
clientsend	send
offline	offline
程式設定檔 (共 1 個 java；1 個 class)	
readinput	analysis
Server 端 (共 5 個 java；7 個 class)	
serverwin	serverwin$1
	serverwin$Button1Mouse1
	Receive
conv	conv
ztof	ztof
decry	decry
trnxml	trnxml

表 3-6java-class 對應表(client 端)

3.4.2 應用系統測試

3.4.2.1 單一元件測試結果(平均時間，單位 秒)

	20kb	500kb	5mb
Eml 轉 Xml	0.3	0.2	1.6
壓縮	0.1	0.8	6.8
加密	0.3	0.4	3.4
Client 介面彈出時間	5	5	5
合併成原來檔案	0.0	0.02	0.9
解密	0.1	0.5	3.8
解壓縮	0.0	0.6	4.1
Xml 轉 Eml	0.4	8	18
Server 介面彈出時間	4	4	4

表 3-7 元件測試結果

Ps：

1. 分析信件此功能因為一直在持續的工作，所以無法用程式估計它的時間。
2. 傳送因為連線的速度不同，也沒有作測試
3. 測試畫面在附錄**頁
4. 系統測試結果留到“第五章 程式執行結果及效能統一“ 作再說明
5. 由以上結論發現，壓縮與解壓縮與檔案的大小比較有正比關係

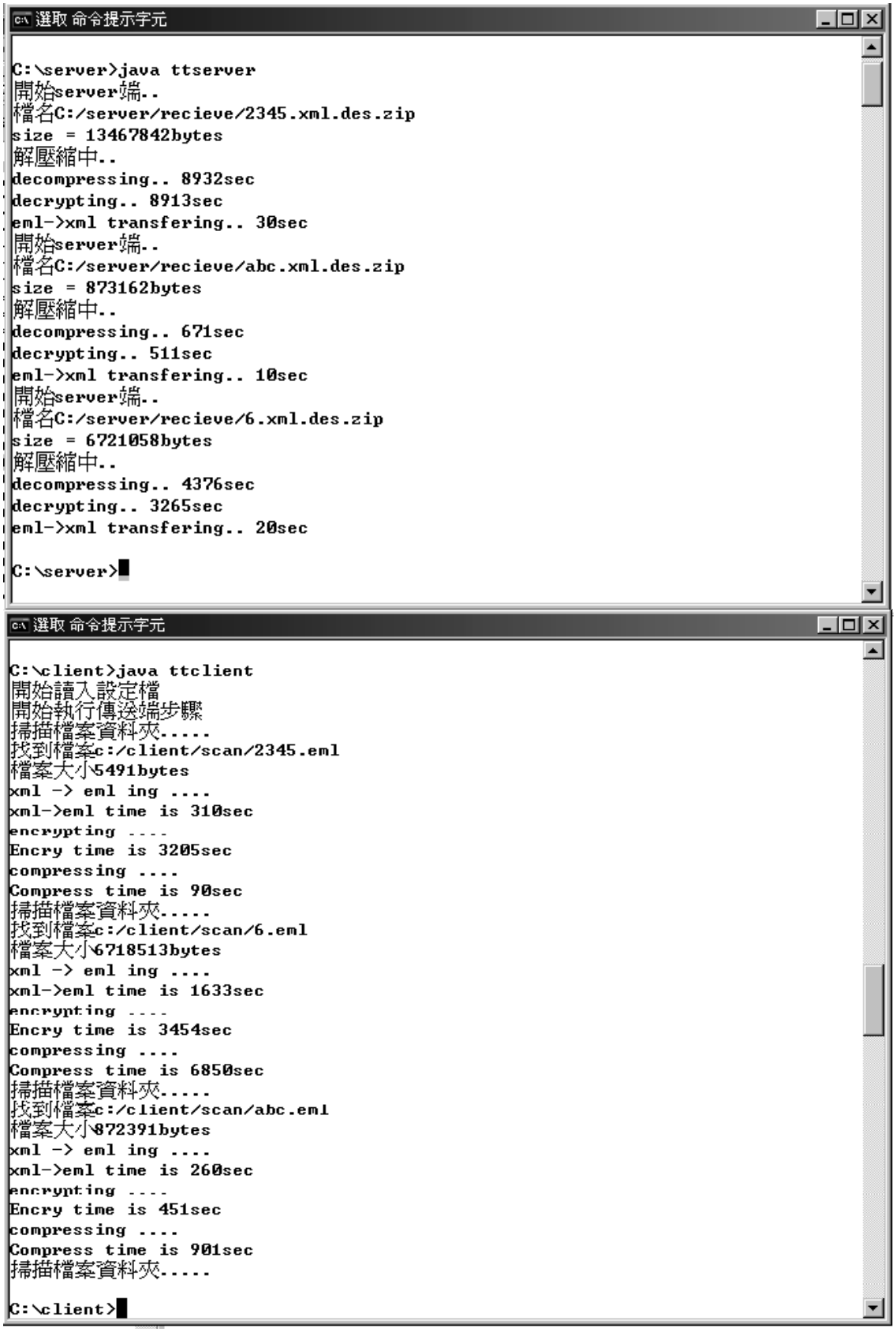

選取 命令提示字元
C:\server>java ttserver
開始server端..
檔名C:/server/recieve/2345.xml.des.zip
size = 13467842bytes
解壓縮中..
decompressing.. 8932sec
decrypting.. 8913sec
eml->xml transfering.. 30sec
開始server端..
檔名C:/server/recieve/abc.xml.des.zip
size = 873162bytes
解壓縮中..
decompressing.. 671sec
decrypting.. 511sec
eml->xml transfering.. 10sec
開始server端..
檔名C:/server/recieve/6.xml.des.zip
size = 6721058bytes
解壓縮中..
decompressing.. 4376sec
decrypting.. 3265sec
eml->xml transfering.. 20sec
C:\server>
選取 命令提示字元
C:\client>java ttclient
開始讀入設定檔
開始執行傳送端步驟
掃描檔案資料夾.....
找到檔案c:/client/scan/2345.eml
檔案大小5491bytes
xml -> eml ing ....
xml->eml time is 310sec
encrypting ....
Encry time is 3205sec
compressing ....
Compress time is 90sec
掃描檔案資料夾.....
找到檔案c:/client/scan/6.eml
檔案大小6718513bytes
xml -> eml ing ....
xml->eml time is 1633sec
encrypting ....
Encry time is 3454sec
compressing ....
Compress time is 6850sec
掃描檔案資料夾.....
找到檔案c:/client/scan/abc.eml
檔案大小872391bytes
xml -> eml ing ....
xml->eml time is 260sec
encrypting ....
Encry time is 451sec
compressing ....
Compress time is 901sec
掃描檔案資料夾.....
C:\client>

圖 3-10 系統測試圖

3.4.3 安裝

3.4.3.1 設定檔內容以及說明

3.4.3.1.1 讀取替代 Domain 之 route.conf ：

route.conf 是用於當 Mail 中寄件地址

Mime tag “To”,”Cc”,”Bcc”

中記錄之郵件地址服務中斷，或網路忙碌時可以由這個設定檔記錄的內容找到替代的寄出位址。

格式說明：

下圖為 route.conf 的範例說明，首先，為下圖 3-11 mail ip 圖中 “example” 中紅色圓圈圍起來的地方，這是郵件地址的格式，可以選擇使用 111.222.333.444 數字 IP 表示法，或用 Domain Name 文字表示法，E-mail 中寄件地址為：

User_Name @ Gateway_Domain_name .mil

而我們在欄位中所填的郵件位址,就是這個 E-mail 加底線的部分。再來是 FORMAT EXAMPLE：

在圖 3-12 是用藍色正方形框起來的補部分,”Gateway” 圖中有加上深藍色粗底線的部分是出現在 E-mail 裡的郵件地址(Gateway)，如果寄出失敗，會找到這個 Domain 而下面的都是可以替代的地址。

最後,是綠色正方形框起來的部分,表示該 Doamin 查詢的解結尾，可以用

”//= = = =” 雙斜線註解，或是”Gateway ：” 下一個待查 Domain 的開始表示分隔，也就是該郵件地址後面沒有更多替代位址了。

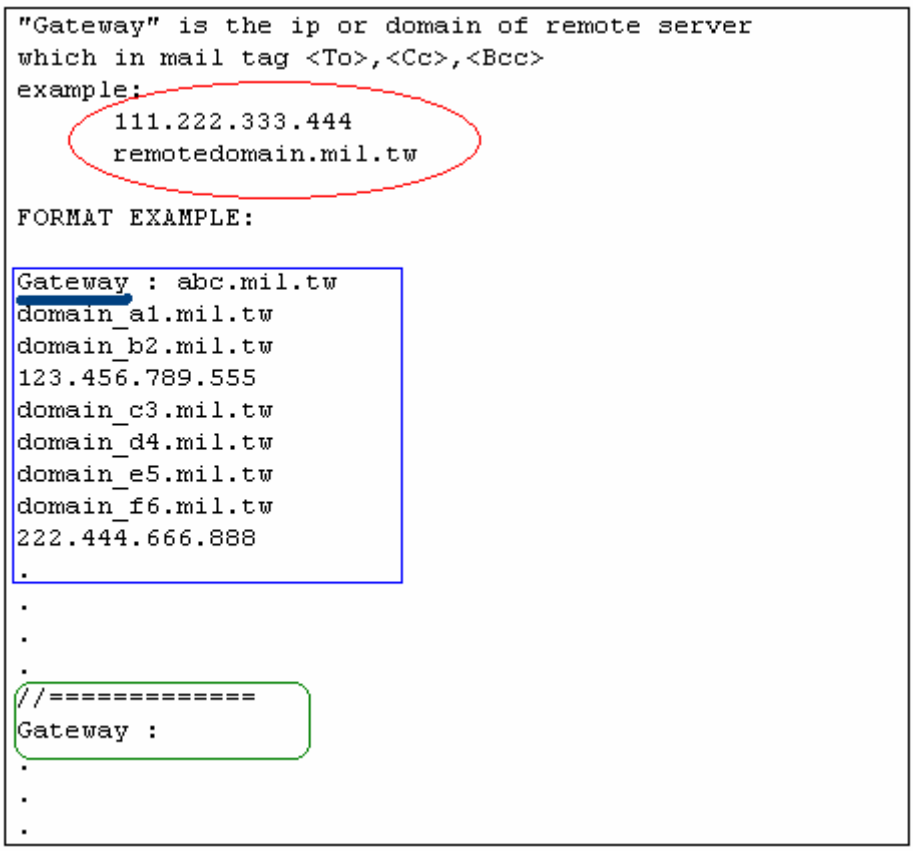


圖 3-11mail ip 圖

圖 3-12 設定檔圖

格式說明：

上圖為設定檔 System.conf 的內容，前面兩的紅色和藍色長方形框，是 Client 和 Server 兩端的使用目錄，在下面 System.conf 參數說明的檔案目錄結構會提到這 5 個目錄的用處。他的格式如圖中所見，在這裡要注意 xml_dir = c:/xml/ 舉 Client 端的 XML 使用目錄當範例，c:/client/xml/ 後面的 /xml/ 為一個目錄，後面要加上斜線。如果有必要修改目錄位址的話，一定要注意，如果沒有設定好，會讓正個程式運作不正常。因為執行的程式會直接把這個當作資料，直接使用。

要注意的項目就是前 5 個目錄設定。

另外，在設定的項目後面就直接加上目錄位置即可，不需要加上雙引號，或括號。

第二個格式要注意的地方就是在設定項目和後面設定的內容之後，也就是整行項目的結尾後面不可以再加文字，也不可以將註解加在後面。註解可以寫在項目的上一行或下一行，

例如：

```
//Client Side Directory
//●可以將註解加在這裡
        xml_dir = c:/client/xml/
//●也可以加在這裡
      temp_dir = c:/xml/client/temp/
//●或是這裡
```

再來是最後一個咖啡色的正方形框，裡面的項目是有關程式運作時需要的一些參數，詳細說明以及設定方法請見下面 System.conf 參數說明，設定項目後面，可以直接寫上設定值，如 scan_tim 後面填時間，

為間格掃描時間；try_time 則是寫次數。

其規定和限制就和上面的目錄一樣，設定項目後面不可以再加註解，如果有必要寫註解以了解項目的內容，可以在其前一行獲後一行加上雙斜線註解 "//" 。

另外，System.conf 的設定檔內容可以用 "//" 雙斜線來當註解。不要使用 /* ~~~~~~~~~~ */ 當註解。

雖然後面這些內容在 readinput.java 中也可以看的到，但是 readinput.java 中的東西 "class conf" 是程式執行中放置變數的地方，除了前四項沒有寫在 system.conf 的以外，後面的幾個項目都不能直接從這裡更改，必須由 system.cong 系統設定檔裡面的 //XML 部分更改，否則設定值無效，每次想要更動設定值，必須在該完後存檔，並且重新啟動程式才可以修改。

3.4.3.1.3 程式內部包含設定： readinput.java

readinput.java 內 "class conf" 的前四項目錄位置設定和 Secure Gateway 其他部分相關，所以不可任意更動，此四項目的內容意義請見下面的檔案目錄結構的參數說明，而後面其他項目，則必須在 System.conf 中設定，在這裡改寫無效。

Readinput.java 和其他 Client 和 Server 部分的程式碼一但 Compile 成 .class ByteCode 後根據下面所述程式目錄設定安裝好後，若沒有程式修改，不可更動目錄位置或檔案名稱。

3.4.3.2 檔案目錄結構

3.4.3.2.1 參數說明

以下前四項目錄設定置於 readinput.java 中，不放在 System.conf
不需要修改

- scan_dir

scan_dir 是放置待處理的 *.eml 檔的目錄，裡面的*.eml 檔會等待 XML 部分的 Client 端程式會掃描這個目錄，將 *.eml 檔讀出，並處理，處理完後刪除。
(※掃描目錄中是否有檔案後，會將之讀取出來，讀取的順序依照目錄中檔名的開頭字母順序，若相同，則選第二個字母順序，以此類推。)

- server_out

server_out 是接收端收到對方閘道傳來的 XML 檔案，經過處理還原回 *.eml 檔後，將處理完的檔案放置在這個目錄，將檔案移交給接收內部信件的其他程式，Server 端程式將ＸＭＬ轉換為 E-Mail 檔放置在這個目錄的期間，會以*.tmp 檔的形式放置，一但檔案建立完成，會更名為*.eml。

- route_conf

放置 route.conf 的位置。
route.conf 是若由信件中 “To”,”Cc”,”Bcc” 讀出的郵件位址若無法傳送，可以在這個檔案內，尋找替代的郵件位址。
Route.conf 詳細讀取以及撰寫格式請見上面！設定檔格式說明

- system_conf

放置設定檔的位置。System.conf 詳細讀取以及撰寫格式請見上面

3.4.3.3 程式目錄結構

Client 傳送端共有 8 個 Java 檔 (檔案內容詳見第一部分，程式功能說明)
將.Java 檔一一 Compile 後，會產生 14 個 .Class 檔(Java ByteCode)

Client.java：
clientwin$1.class
clientwin$Button1Mouse1.class
scan.java：
filescan.class
remind.class
NameFilter.class
mailtoxml.java：
eml_reader.class
emltoxml.class
addfile.class
remote.java：
sendto.class
encry：
encry.class
ftoz.java：
ftoz.class
clientsned.java：
send.class
oddline.java：
offline.class
second_domain.class

Server 端則共有 5 個 .Java 檔，有 7 個 .class 檔：

serverwin.java：
- serverwin$1.class
- serverwin$Button1Mouse1.class
- receive.class

conv.java：
- conv.class

ztof.java：
- ztof.class

decry.java：
- decry.class

trnxml.java：
- trnxml.class

另外，還有一個程式檔：readinput.java 用來讀取設定檔 system.conf 的，因為 Server 端的程式和 Client 端的程式都會用到讀取設定檔，所以將 readinput.java compile 後所產生的三個 .class 檔：

conf.class，conf_reader ，analysis

各放入 Server 端和 Client 端的執行檔目錄中：

Server：c:/Server

Client：c:/client

3.4.3.4 XML JDOM & SAX Pakage 的安裝：

程式將 *.eml 轉為 *.xml 是用 JDOM 的 API 套件做的，所以程式執行的時候必須在 Java classpath 裡面加入幾個檔案，以供轉換使用，另外，如果 Gateway 的 JDK 裝 1.3 版的話，加密也必須用到相關的 API ，所以也必須將一些 package 加入到 Java 預設 classpath 裡，要加入以下的 .jar 檔：

- ant.jar
- collections.jar
- crimson.jar
- jaxp.jar
- jce1_2_2.jar
- jdom.jar
- jdom-jdk11.jar
- local_policy.jar
- US_export_policy.jar
- xalan.jar
- xerces.jar

共 11 個 .jar 檔。

要將這 11 個 .jar 放置於 ：

C:\ ProgramFiles\ JavaSoft\ JRE\ 1.3.1_09\ lib\ ext
C:\ jdk1.3.1_09\ jre\ lib\ ext

(兩個 java JDK 安裝目錄) 下，才可順利執行轉換 Email 到 XML 的工作。

3.4.4 使用方法：

當檔案和程式都依照上面的說明安裝好了之後，就可以執行了。執行方法有兩個步驟：

- Client 端的程式執行：

進去 c:/client/
執行 java clientwin 即可。

- Server 端的程式執行：

進去 c:/Server/
執行 java serverwin 即可。

有一點要注意的事，Server 端和 Client 端的程式都會共同用到 System.conf 所以兩在執行上述兩個程式時，應該相隔 5 秒鍾，兩個程式讀取 System.conf 的時間隔開。

第四章系統開發與實作

4.1 基本架構：

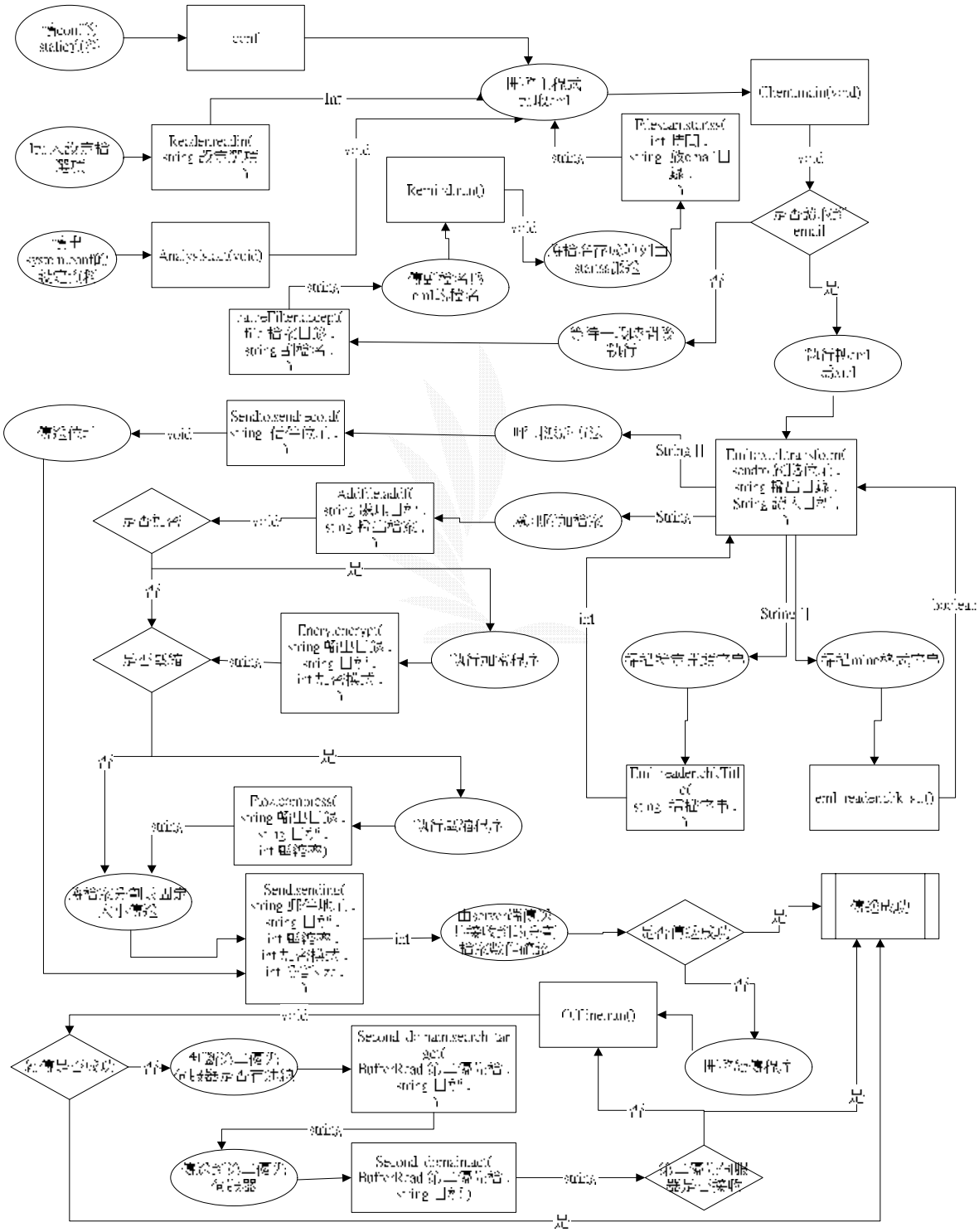


圖 4-1client 端 函式之間關係圖

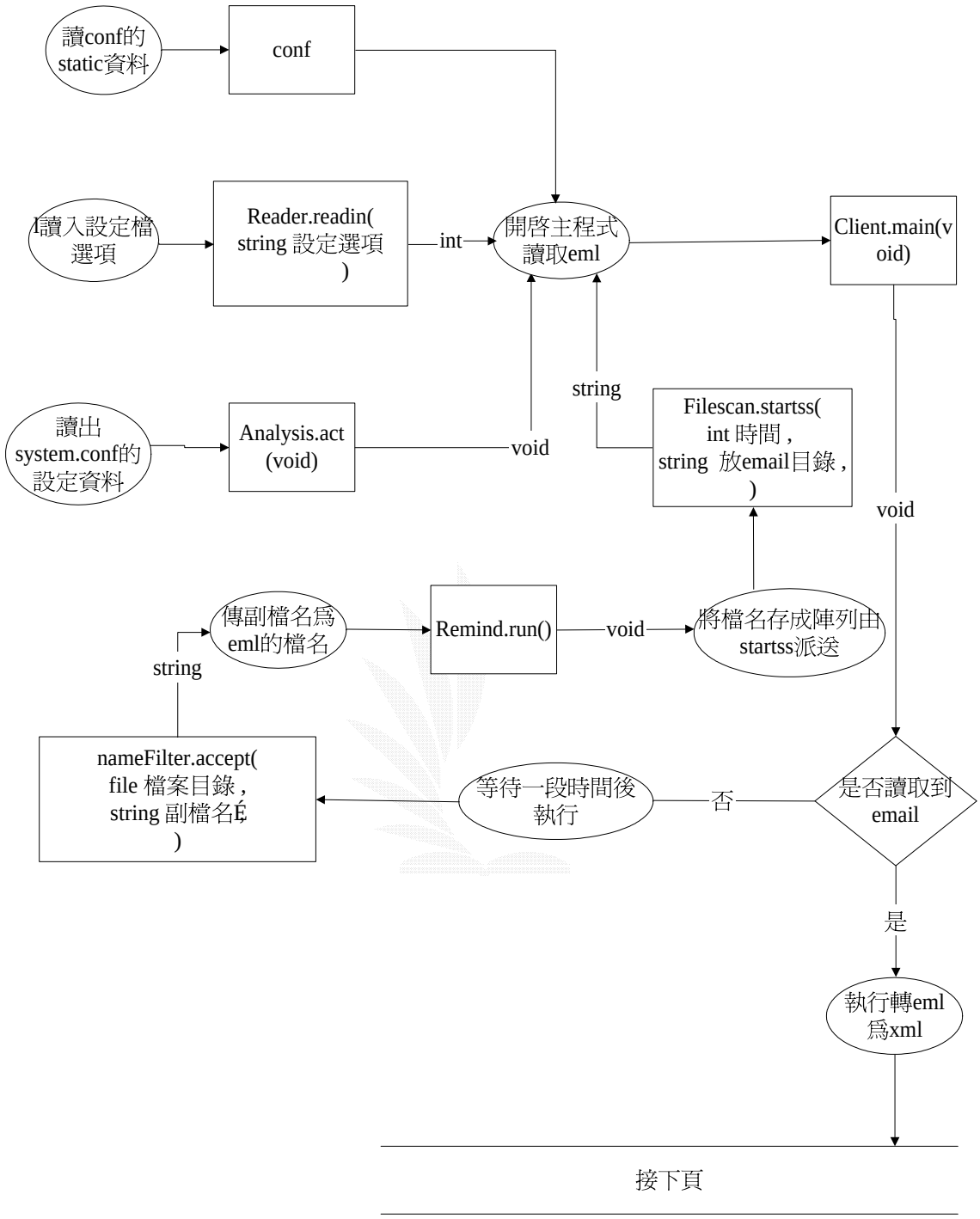


圖 4-2client 端分解圖

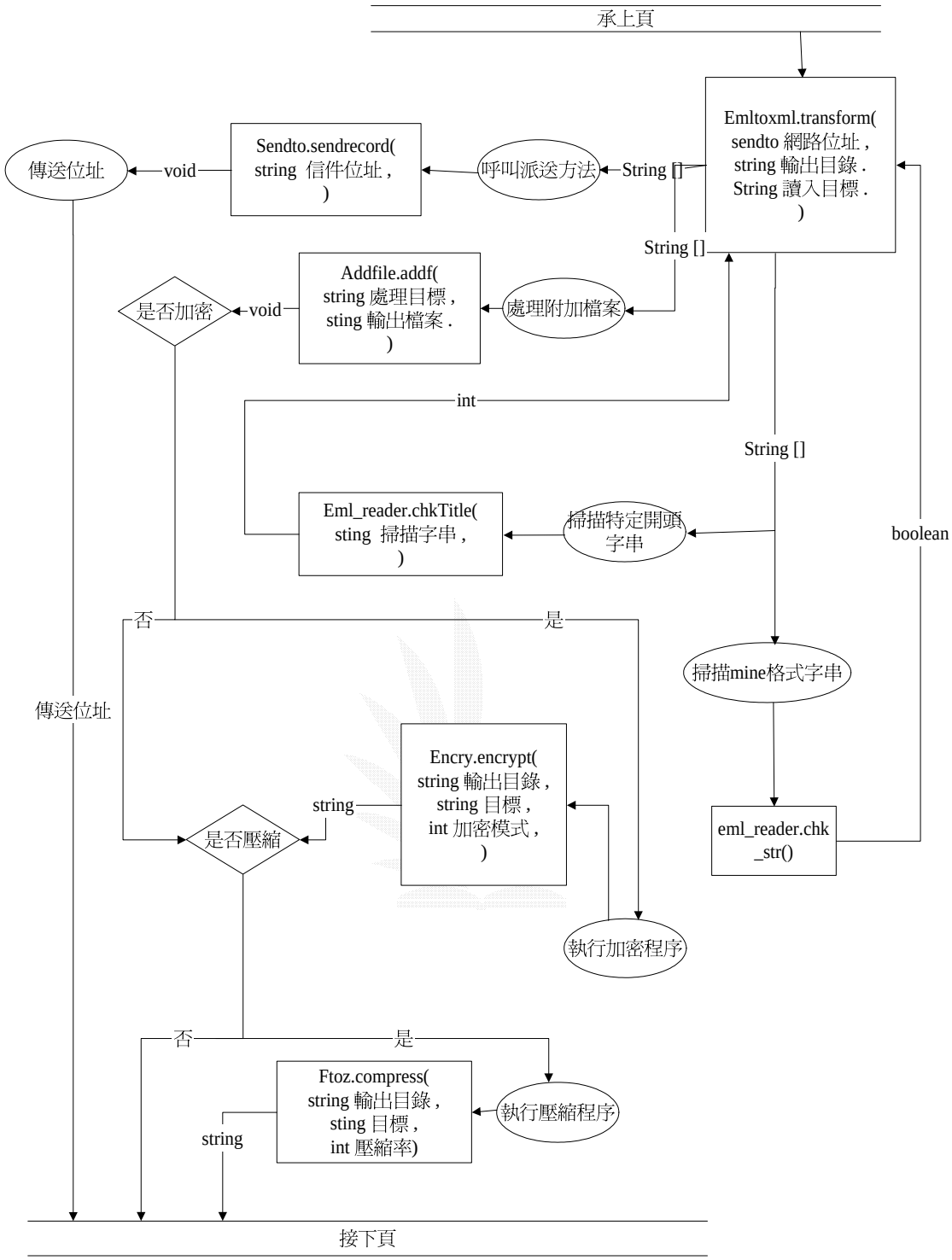


圖 4-3client 端分解圖

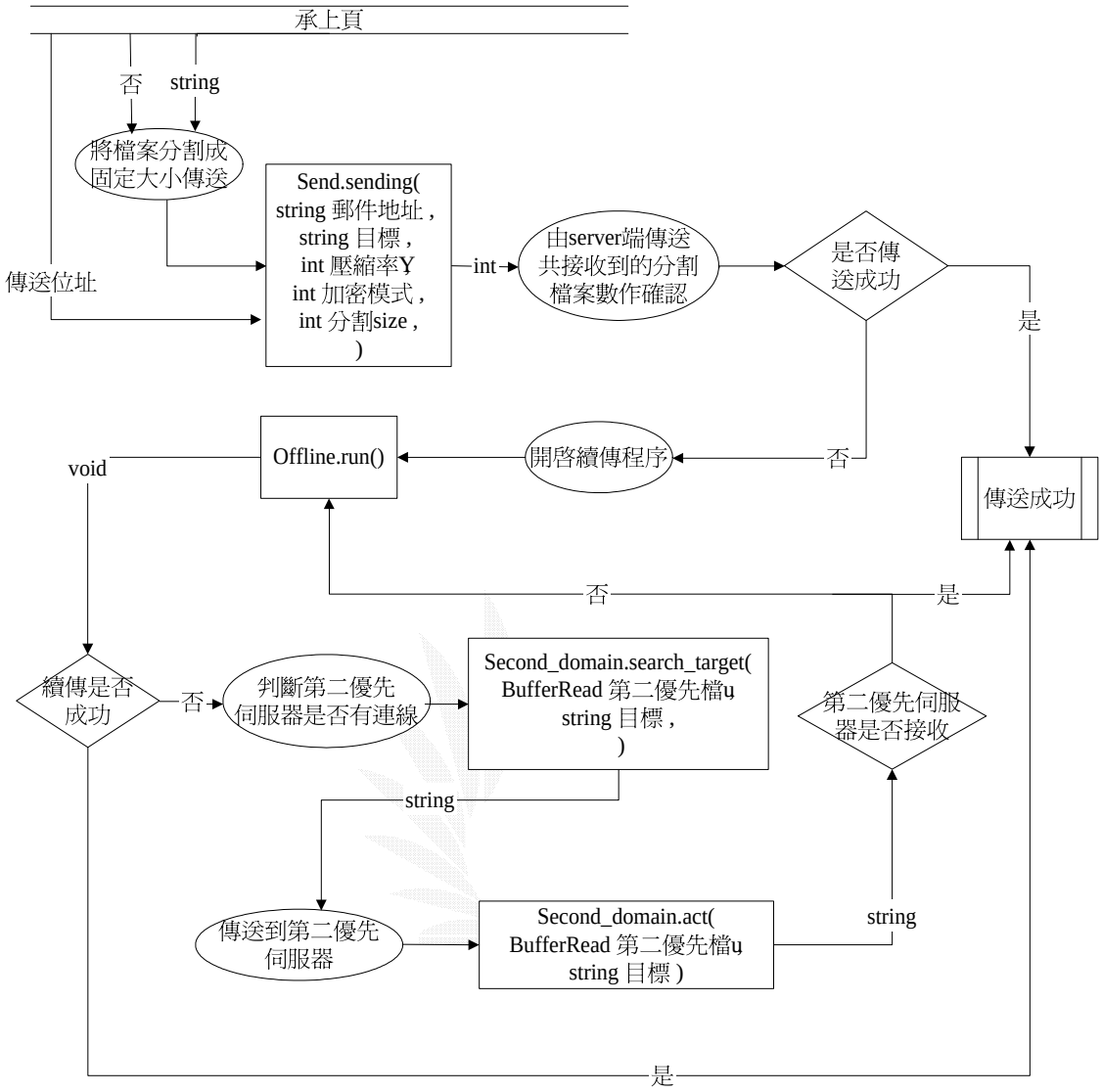


圖 4-4client 端分解圖(下段)

根據前面的評估以及分析，這裡我們需要的功能有：

- 讀取設定檔 System.conf、
- 掃描目錄(固定時間) 、
- 一個可存放設定值的記憶體空間、
- XML 建立和轉換、
- 讀取壓縮以及加密等參數、
- 讀取 MIME 格式內判斷以及記錄送出位址的函式、

- 加密、
- 壓縮、
- TCP/IP Socket 檔案傳送、
- 在一定的失敗次數內可以提供斷點續傳(配合檔案分割)、
- 讀取傳送失敗後可以選擇替代位址的 Route.conf
- 以及一個主控流程的主程式。

除了這些功能之外，還需要一個暫存處理目錄，”/temp”，這些功能將組合成 Client 端的全部功能，每個功能都以一個 Class 的形式，然後裡面包含該 Class 會到的 Method(方法)，然後有一個主程式 Main 統合這些功能。

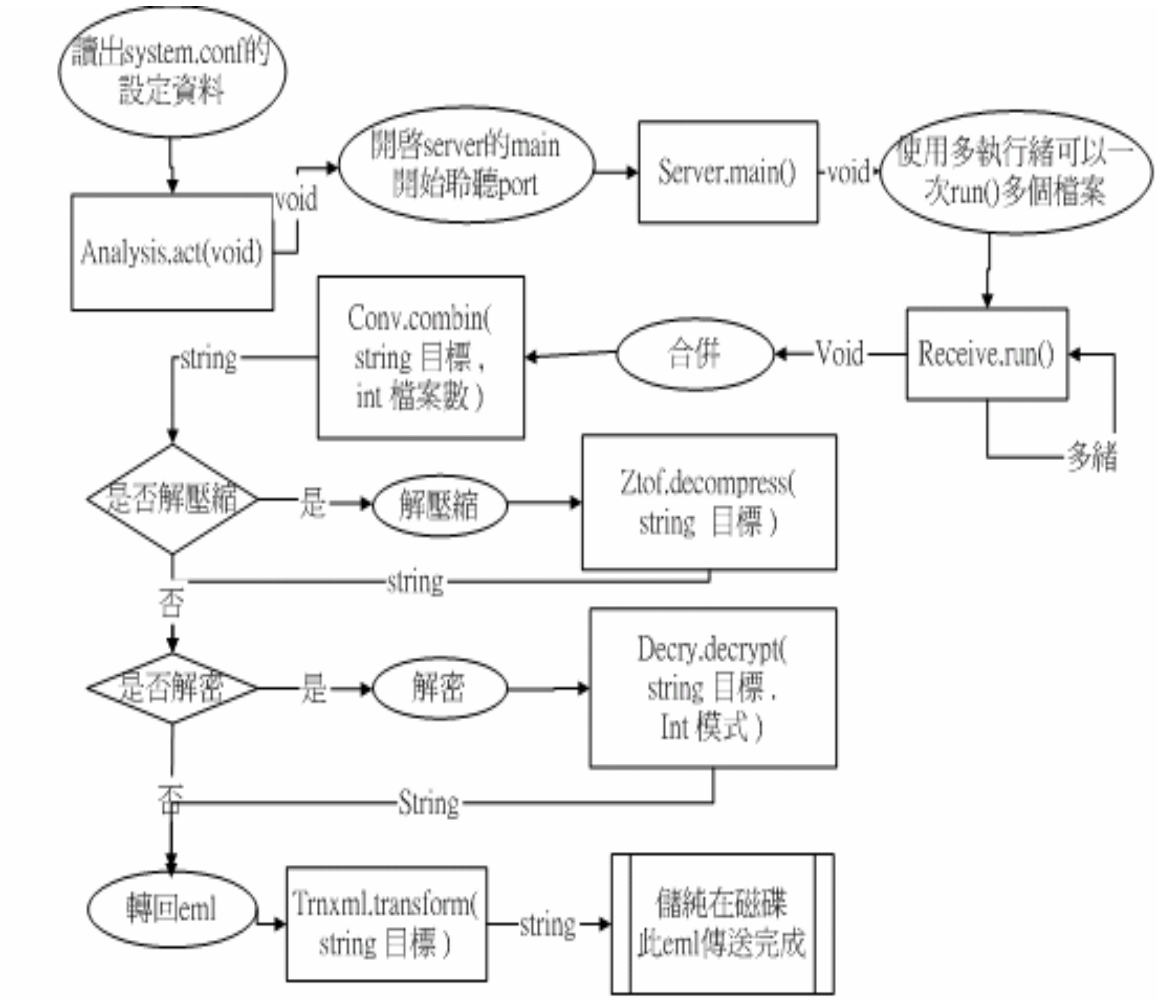


圖 4-5server 端 函式之間關係圖

而 Server 端根據評估，只須被動的根據 Client 端所給的資訊對接收的檔案做：

- 組合(根據檔案數目、大小)、
- 解壓縮、
- 解密、
- 傳給 XML 處理目錄、
- 還原回 MIME 格式給用戶。

除了這些功能之外，還有兩個提供暫存的目錄，”/temp”,”/recieve”其中，receive 是用於接收分割的檔案，再將之組合。

4.1.1 Client 端(傳送)的主控程式(main)

根據系統分析的結果，Client 端主程式為

控制 Client 端 -> 轉 XML -> 加密 -> 壓縮 -> 傳送。

流程的控制，另外，此程式為常駐程式，在執行以後會進入 while( ) 迴圈中不斷的重複執行，在迴圈中會用到的物件實體都是固定的，所以在程式執行後，先建立 Client 主程式會用到的變數以及物件實體，建立完成後再進入 while( )迴圈中執行。

另外，主程式裡面的方法或實體內的函式程式必須遵守固定的參數格式，例如下圖 4-6. 所示，目的是讓主程式的參數引用格式化。未來果需要擴充主程式較為方便，只要遵守這個規則即可達成，要維護修改也非常容易，而該函式如果有特有的額外參數，可以加在後面，例如：ZIP 的壓縮率、加密的演算法選擇。而參數的使用在刪除暫存檔會有額外的說明。

另外，client 端主程式必須根據轉換 XML 所提供的是否壓縮、加密兩個 Flag Bits 給與 XML 格式的 E-Mail 做適當的處理，在最後要判斷程式是否完成，來決定要不要刪除以處理過的 MIME 格式的 E-Mail 檔。

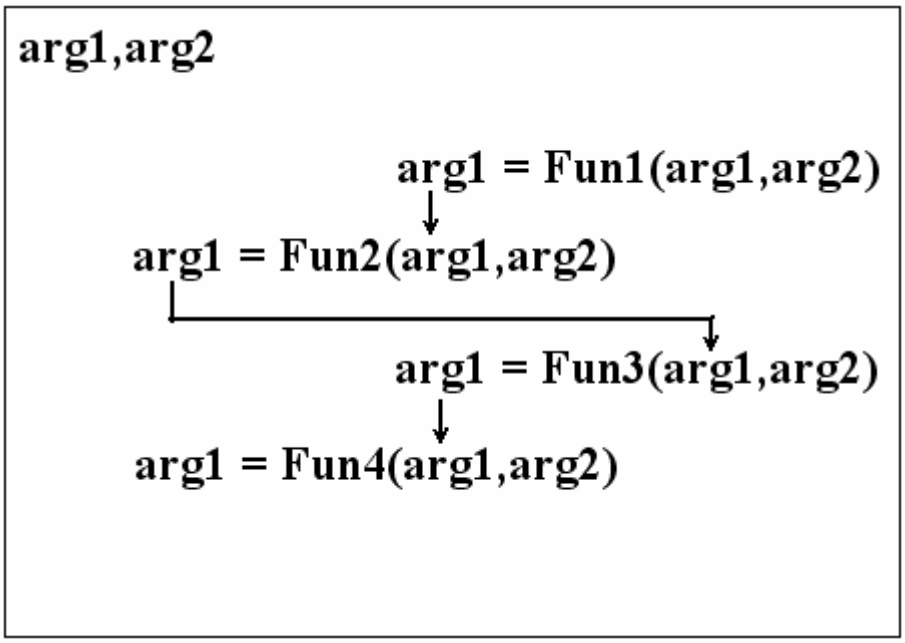


圖 4-6 Client 端 Main Fcunction 參數引用示意圖

4.1.2 Server 端(接收)主控程式(main)

Server 端的主程式也有個 while( )迴圈，也是常駐程式，但是和 Client 端的程式有很大的差異。

根據系統分析的結果 Sever 端的接收是完全被動的根據 Client 端的要求接收檔案，而測試對方是否可以傳遞檔案、檔案資訊、或續傳的一切處理行為都是 Client 端在做，所以 Server 端的工作單純化就達成了，而工作內容有：組合(根據檔案數目、大小)、解壓縮、解密、傳給 XML 處理目錄、還原回 MIME 格式給用戶，這一連串流程式的工作模式下，參數的使用還是必須跟據圖 4-6 所述的方法。參數的使用在刪除暫存檔會有額外的說明。

其主要的設計概念就是常駐，聽伺服器的某個 port，如果有接到傳送

要求，就開一個 Socket 並將傳給一個執行緒(Thread)去接這個檔案，Thread 裡面做完所有的事(解壓縮、解密等)，後結束。

Server 如果 Accept 到 Client 端的請求後，會用 TCP/IP 的方法建立連線，下圖 4-7 為連線的意示。詳細的連線方法見 Client 端的 TCP Sccket 連線、檔案傳送。

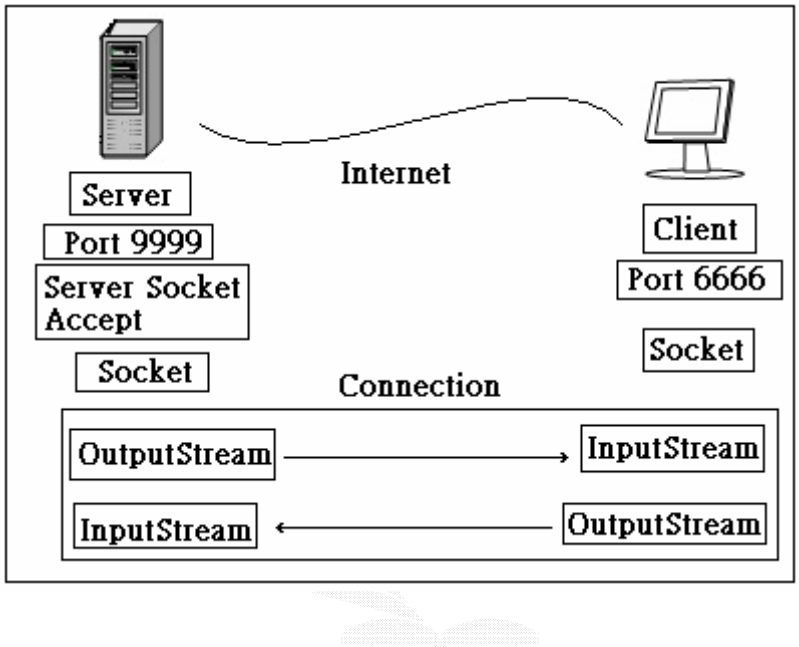


圖 4-7Server、Client 兩端傳送意示圖

如下圖 4-8 所示，new 一個 Thread 來處理 client 的 request 就是接收 client 端的檔案，根據訊息加以處理。訊息也是 client 端傳送的，唯一段字串資訊，根據系統分析的結果，其格式如下所示：

The Information Format

Example:

"21,0,1,name,13"

21：filecount

0 : if Compress or Compress rate

1 : if Encry or Encry mode

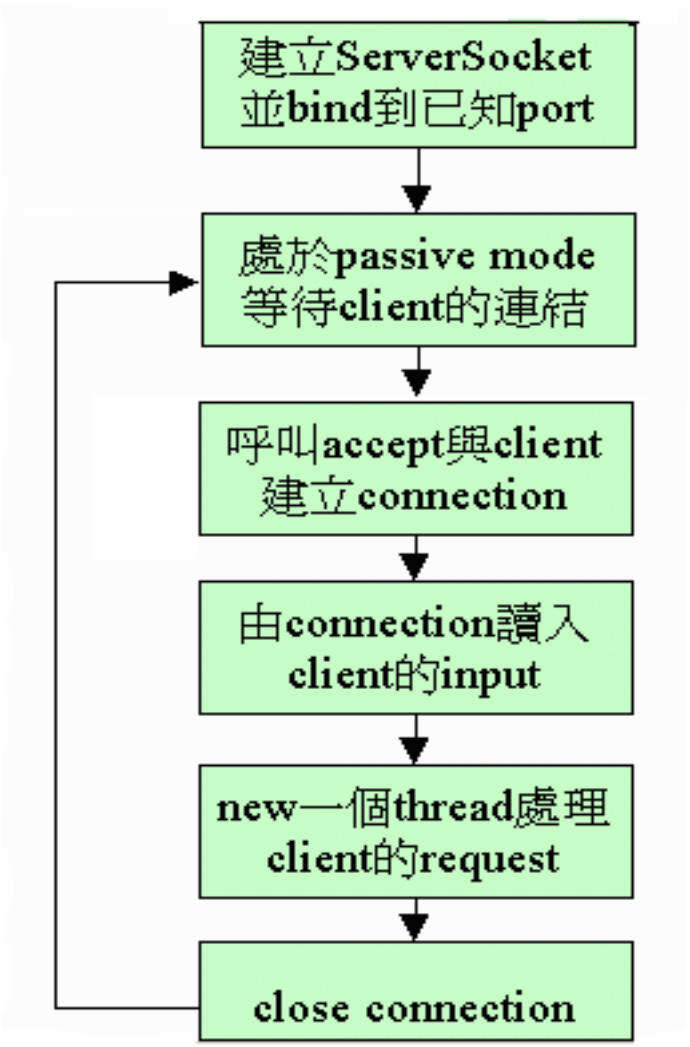


圖 4-8Server 端接收檔案處理流程

4.1.3 Client、Server 兩端暫存檔刪除處理

除了 Client 端必須遵守圖 4-6 所表示的參數規則外，還有一點必須在實作上注意，就是暫存檔用完後的刪除，我們在 Server 端和 Client 兩端都有可以存放暫存檔的目錄，只要每個會用到並建立處理過程中間的暫存檔，在程式結束時，只要確定已經完成目前的工作後，即可把用過並且不需要在用到的站存檔刪除，如下圖 4-9(圖 4-9 只是表示程式運作時會有的讀檔、建檔和刪檔傳來的檔案有時候並不需要解壓縮或解密) 所示。

當 Server 接收到 Client 要求會建立一個 Thread 收傳來的檔案，組合好後，檔案控制權會交給解壓縮，解壓縮從組合好的那個檔案讀出 (藍線)，並加以解壓縮，解壓縮完成後會在一個暫存目錄建立解完壓縮的檔案(黑線)，當使用該檔程式確定整個程式做完工作後，會刪除原來的檔案(紅線)。

圖 4-9 只有講到 Server 端的部分，其他在 Client 端的讀取 E-Mail、轉 XML、將檔案加密和壓縮到傳送都是用這種概念。

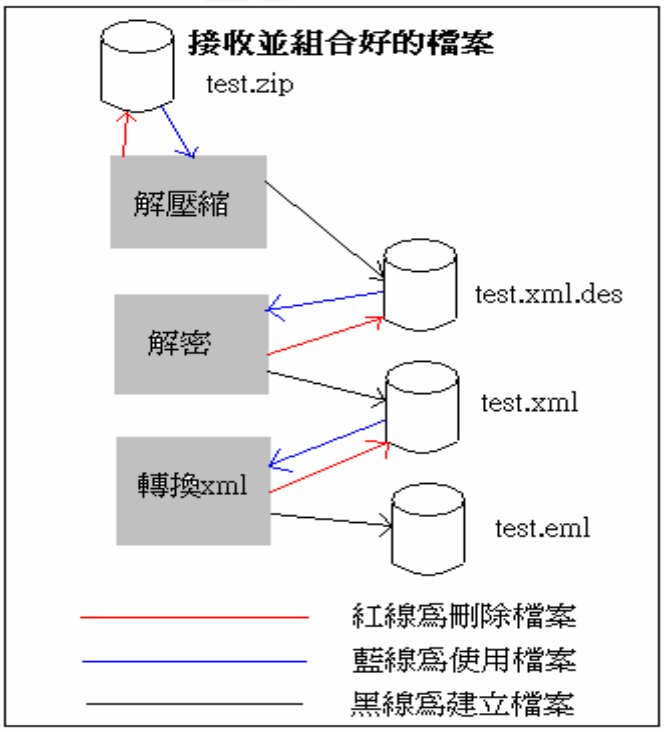


圖 4-9 處理完後刪除用過的檔案，及檔名變化

最後，圖 4-9 還說明了一件事，我們在系統分析的時候，有討論到這個地方，當檔案在處理過程中，副檔名的問題，副檔名代表了現在者個檔案內容是什麼，該怎麼處理，而且可以分別現在處理到什麼種的檔案，所以以 Client 端完整的檔名變化應該是：

test.eml -> test.xml -> test.xml.des ->test.xml.des.zip

而在 Server 端的檔名變化則可以見圖上的變化。

4.2 讀取設定檔：

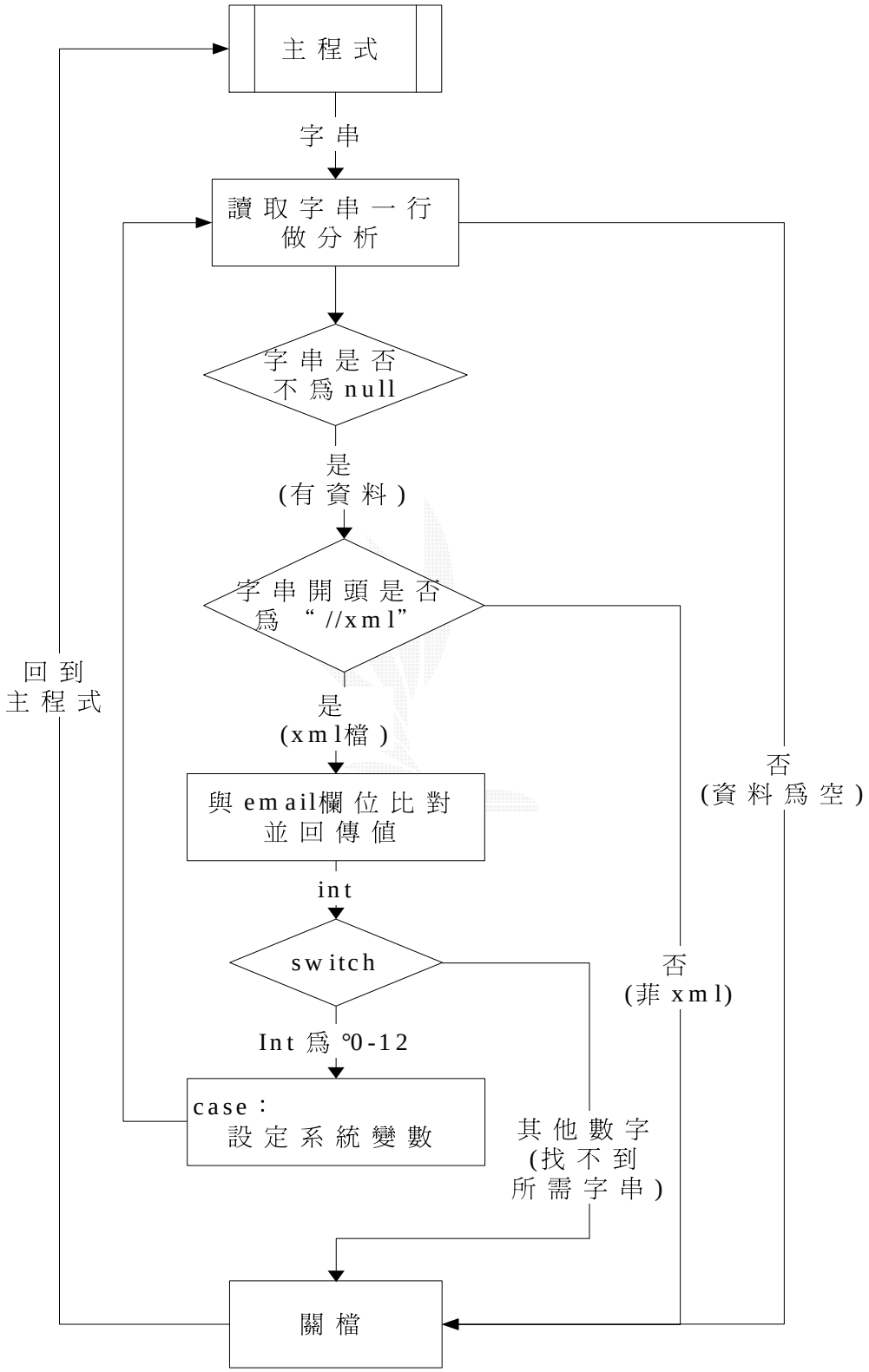


圖 4-10 如何讀取意示圖

整個 XML 傳送程式有個 System.conf 記錄一些運作時需要用到的選項：

設定檔內項目	其代表的意義
divide_size	續傳時檔案切割大小
Scan_time	掃描目錄時間
buffer_size	程式運作的 Buffer
remote_port	接收端的 port number
Send_port	傳送端的 port_number
Try_times	傳送錯誤時嘗試次數
Delay_time	每次錯誤後停止時間

表 4-1conf 選項表

利用 Java.io 內的 BufferedReader 開啟設定檔，上面的項目都一定會在文字檔內一行的開頭，所以當我們讀取到一行的開頭為所需要的字串時，將它讀入預先建立的一個物件實體 Class conf，都是廣域的變數，能讓運作的程式可以取用。

4.3 掃描目錄

由於 Client 端在接收處理 Mail 的方式是將檔案丟到固定的目錄，以等待處理，所以必須有一個程式可以掃描目錄傳回目錄裡面有哪些檔案。

設定檔裡有個選項 scan_time 就是用在讓 scan 目錄的程式可以格一段時間掃描一次目錄，這裡我們用的是 Java 的 class File，裡面有個方法 **list( )** 可以將某個目錄中所有檔案加以傳回。

Scan 目錄用到的”¨class NameFilter implements FilenameFilter” 這個 FileName Filter 是專門用來過濾檔案名稱

的，我們用它來過濾掃描的目錄中的檔案的附檔名是 .eml(表示郵件)的檔案。而掃描到的檔案會給主程式開始做處理。

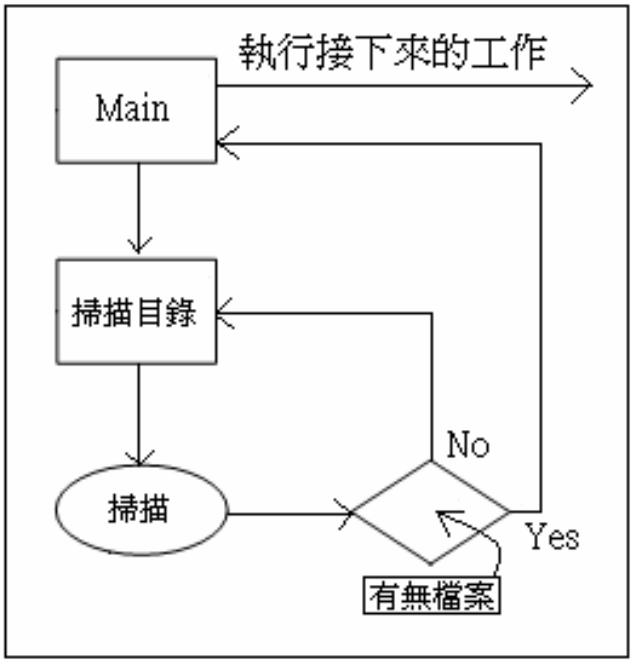


圖 4-11 掃描目錄的工作過程

另外，這個系統的需求有一樣是提供信件優先權的功能，根據系統分析的結果，我們傳送郵件需要有優先權，這個目的可藉由這裡達成，我們可以將目錄中的檔案以名稱的順序排序，

例如：

0_abc.eml 的順序會比 a_abc.eml 的順序還要前面。

如此，掃描出的檔案放在陣列後，可以用這種規則加以排序，在陣列越前面的檔案會越先送，以這種方法可以在這掃描階段就達成傳送優先等級。

4.4 XML 轉換

在這個系統中，將信件直接轉換為 XML 格式，並以 XML 的形式處理，轉換完成後的檔案可以提供日後自己 client 端的處理或是傳給接收 server 端後的處理，例如，使用者可以自行擴充將信件以 XML tag 建檔或建立資料庫或分類，此為以 XML 做為每個 Gateway 之間檔案的傳遞的目的。

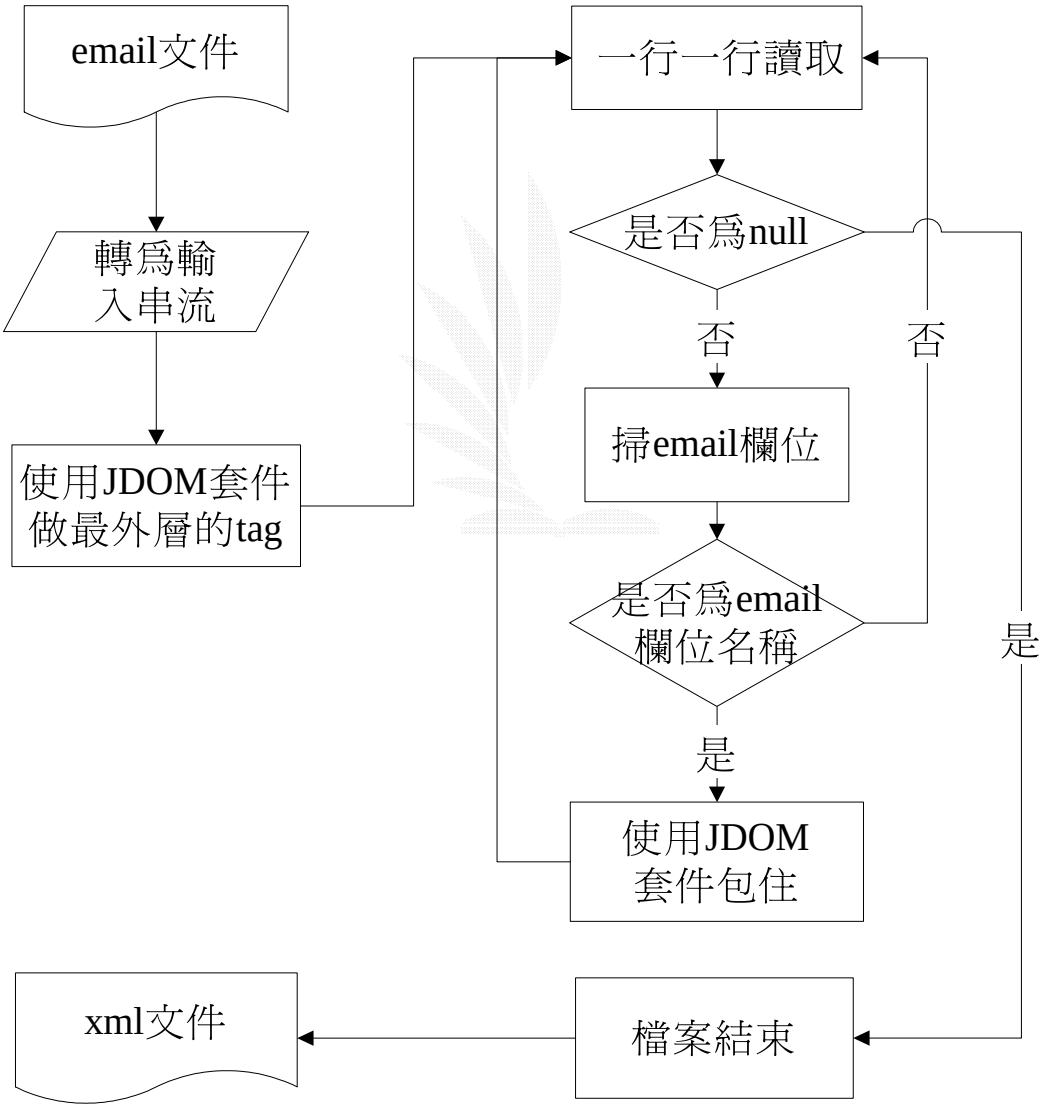


圖 4-12xml 運作意示圖

將對以下的 MIME tag 做 XML 的轉換工作：

MIME tag	代表意義
1. Received	轉寄地址
2. From:	寄出者
3. To:	傳送地址
4. Cc:	附件
5. Bcc:	秘密附件
4. Subject:	主旨
7. Date:	建立日期
8. Size:	大小(附加)
9. MIME-Version	MIME 版本
10. Body	內文
11. Attach	附加檔

表 4-2 已與系統分析輸入部分重複表

我們採用 JDOM 來轉換 E-Mail 為 XML，根據系統分析的結果，JDOM(Document Object Model)以及 SAX(Simple API for XML) 我們選擇 JDOM 轉換 XML，因為其以物件式的處理模式，較適合我們在轉換過程中加入一些額外的處理功能。

以下為掃描 email 欄位圖，掃出所需要的 email 欄位，並傳出值：

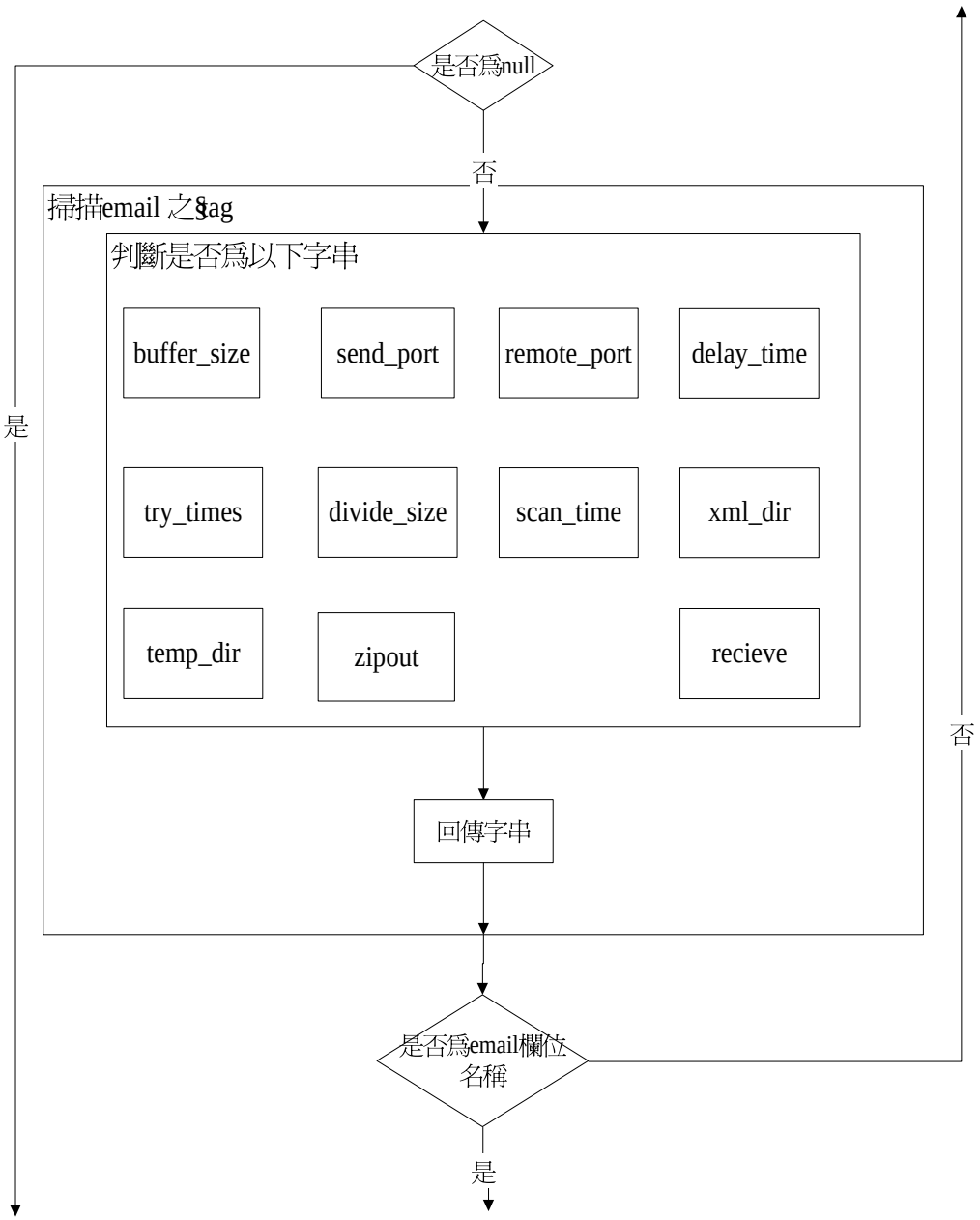


圖 4-13 掃描 email 欄位圖

另外，我們之前研究 XML，我們知道有 DOM(Document Object Model)這個 API，我們知道她的缺點是太過佔用記憶體資源，但是 JDOM 卻是針對 Java 2 開發者設計的，擁有像 SAX 般的循序效能，也有如 DOM 一樣，可隨機操作 XML 文件。

我們所需要的額外處理功能就是必須要在 MIME 格式中的”To”,”Cc”,”Bcc”等三個 tag 中讀出送出位址，因為我們利用 JDOM 的每個子節點物件化的特性，在每個建立 tag 的操作過程中，插入記錄位址的”record”此為動態陣列 Vector：

```
static Vector position = new Vector();
```

下圖 4-14 為取出目標郵件地址的流程圖。

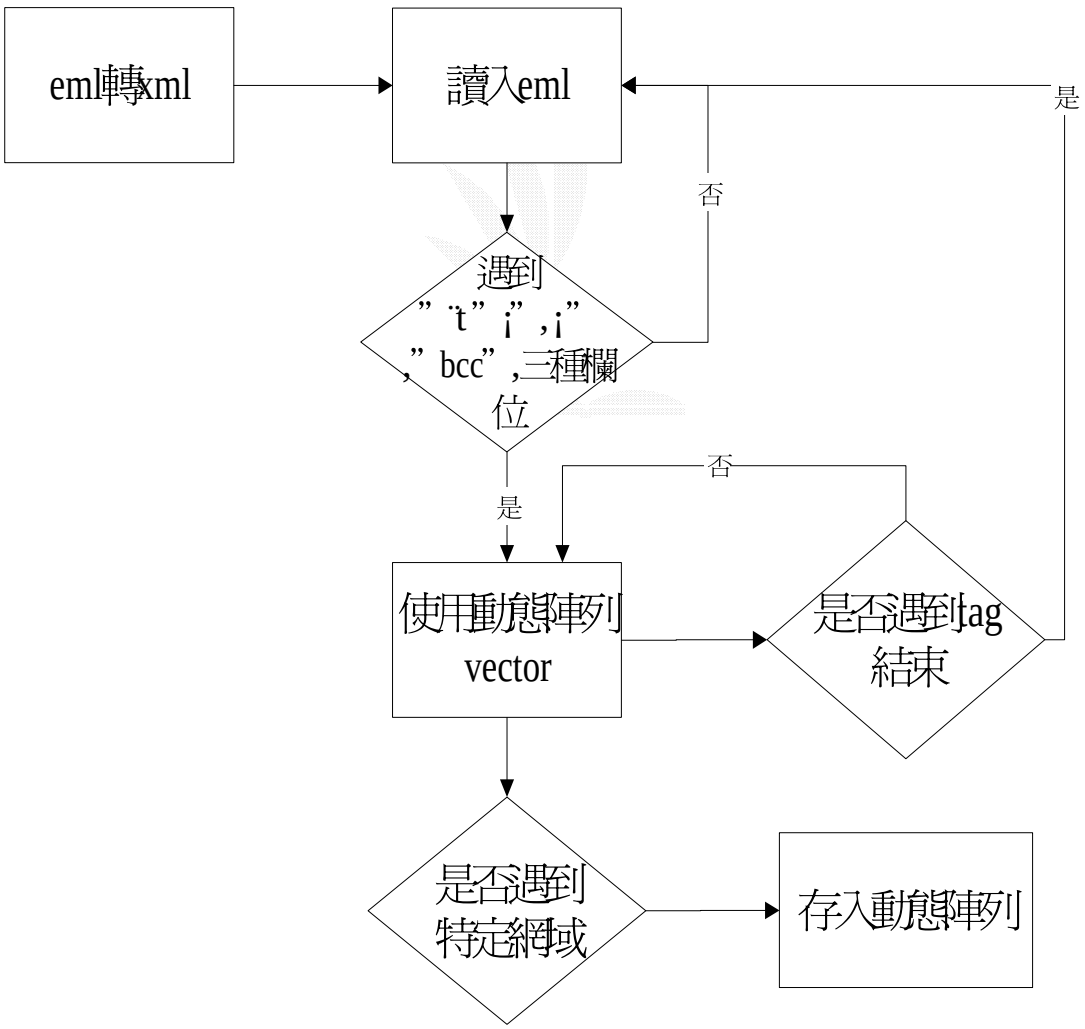


圖 4-14 取出送出位址流程

在”To”,”Cc”,”Bcc”裡面的郵件地址我們都會送到一個 Check 函式去檢查他的位址,根據系統分析我們這個程式的目的並不是通用的 Mail Server，目的是選擇固定的網域傳送，例如：

140．134．???．??? ->

如果在判斷地址的程式裡面的判斷字傳設定為”140.134.”的話，所有逢甲大學的地址都會被選擇

mail.a_instance.com.tw->如果在判斷地址的程式裡面的判斷字傳設定為

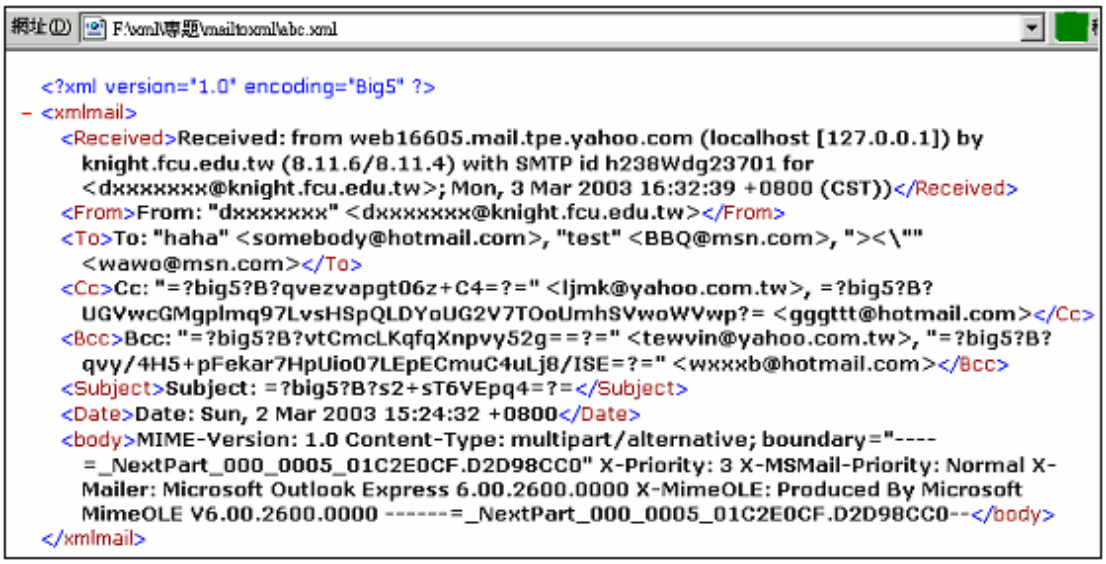
”mail.a_instance.”

則所有郵件地址裡面，包含該字串的都會被提取。

建立 XML 文件：

```
Element rootElement = new Element("xmlmail");
Document document = new Document(rootElement);
```

第一行，建立 XML 文件的根節點，跟目錄是每個 XML 文件所必須的節點，第二行是 JDOM 的重點 Document 這個物件所建立的實體可以提供我們建立子節點或是子節點的子節點的依據。



XML 對 MIMI 格式輸出結果

我們可以對 Document 輸出我們想要的 XML 格式，再將其交給 XMLOutputter 建立檔案：

輸出的檔案就是 XML 的處理結果。

```
XMLOutputter outputter = new XMLOutputter();
```

其次，我們可能會將 XML 還原回 E-Mail 的 MIME 格式，因為這封信可能會被轉寄，或是會以信件的格式被使用者閱讀。所以這裡有將其轉回 MIME 格式的功能。

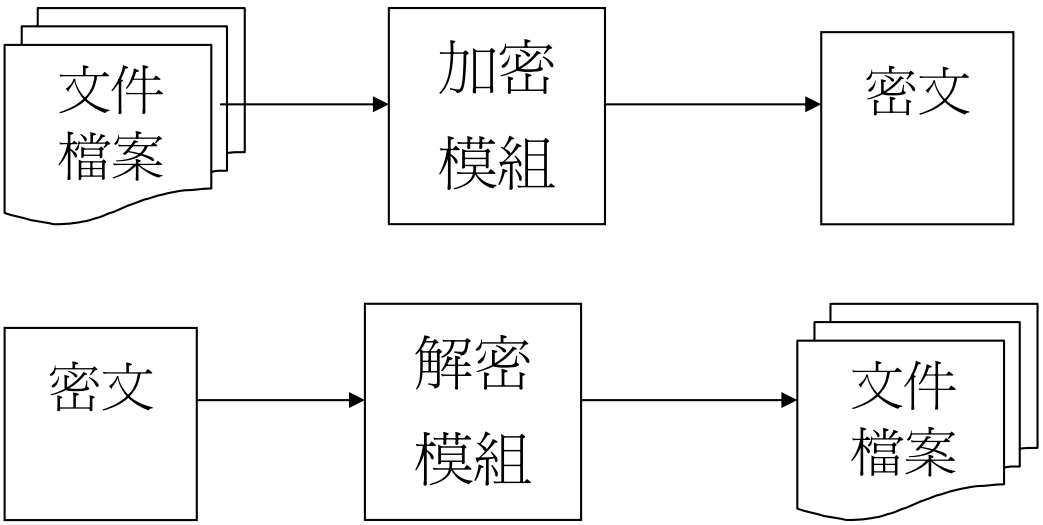
我們轉回 MIME 格式有兩種種選擇，一個是用 SAX(Simple API for XML)依照整個 XML 的順序讀入 SAX 解析器，依照 tag 的順序將 XML 的 tag 去掉，還原為 E-Mail 的格式。另一個方法是直接從附加檔直解拿出整封信件，因為我們在 Client 端建置 XML 文件時，直接把整封信塞入附加檔。

4.5 加解密功能

java 無論是 1.4 以上直接內建 jce 的 api 或 1.3 版以下要額外安裝 jce，實做加解密都同樣載入以下套件

```
import java.io.*;
import java.security.*;
import java.security.spec.*;
import javax.crypto.*;
import javax.crypto.spec.*;
```

關於密碼的濾器串流其主要的部分都放置在 java.security 套件中，並且提供了兩種類別，分別為摘要輸入串流和摘要輸出串流，而摘要串流利用的是 javax.crypto 套件中，java.io.*則是基本檔案處理套件。



4.5.1 加密

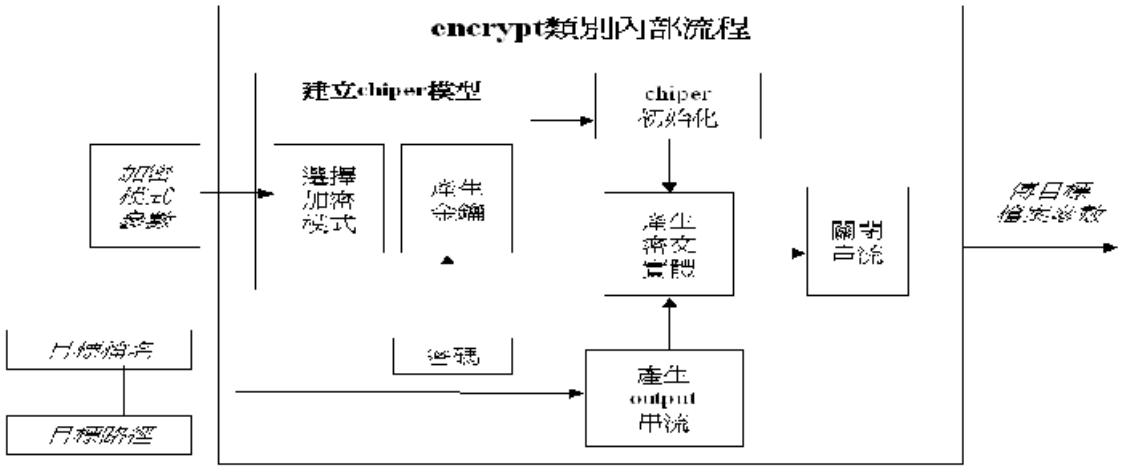


圖 4-15 加密流程圖

加密設計步驟：

1.將愈加密文件輸入至輸入串流

2.建立一個 key

- 宣告 DESKeyFactory 物件
- 宣告 SecretKeyFactory 物件
- 宣告 SecretKey 物件
- 呼叫 SecretKeyFactory 的 getIntance()函式

3.宣告 Cipher 物件

4.初始化 Cipher 物件

- 呼叫 cipher 的 getInstance()函式
- 呼叫 cipher 的 init()函式

5.將 Cipher 轉成一個濾器

- 宣告 byte[]陣列來接 chiper 的位元向量

6.讀取 ”輸入串流” 經過 “加密濾器” 成 “輸出串流”

7.關檔

在 java 中文件要進行加密，簡單來說就是先建一的要加密的陣列向量，然後預加密的文件分成很多小段，每一小段印射入這個陣列向量而產生出來的資料，轉入到輸出串流中，所形成的檔案即是我們的密文，加密流程圖如上。

4.5.2 解密

解密的概念跟加密很類似，先要讓解密模組知道所需要的參數，如密碼、模式、padding 的方法等等，才能夠成功解出密碼

解密設計步驟：

1.收詢到加密文件輸入至輸入串流
2.建立一個 key
- 宣告 DESKeyFactory 物件
- 宣告 SecretKeyFactory 物件
- 呼叫 SecretKeyFactory 的 getInstance()函式
- 宣告 SecretKey 物件
- 呼叫 SecretKeyFactory 的 generateSecret()函式

3.讀取初始化項目
4.使用加密標準及初使化
- 呼叫 cipher 的 getInstance()函式
- 呼叫 cipher 的 init()函式

5.轉成一個解密濾器
- 宣告 byte[]陣列來接 chiper 的位元向量

4.讀取 ”輸入串流” 經過 “解密濾器” 成 “輸出串流”
7.關檔

加密與解密的步驟差不多，不過在建立 key 的時候，需要多執行 SecretKeyFactory . generateSecret()函式，其他的程式碼與加密有對稱關係

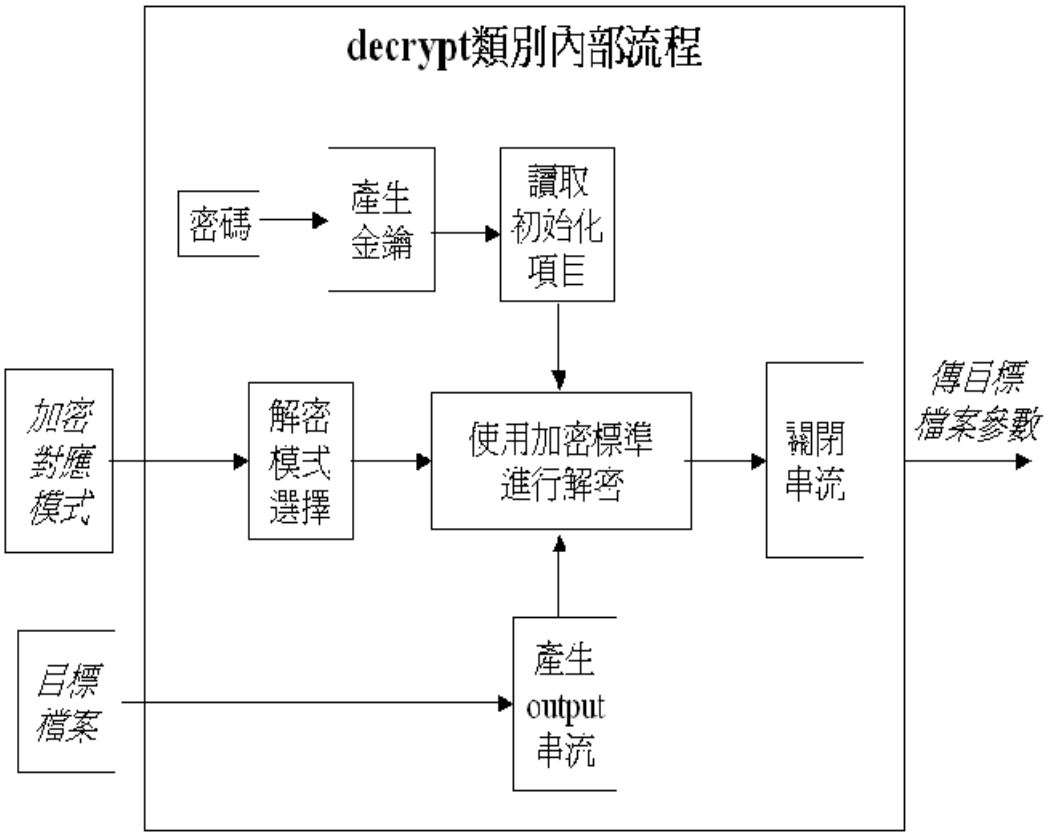


圖 4-16 解密流程圖

加密所需要的重要資料 key 值，由於是 des 的加密方法，所以我們將他設計在程式碼之內，至於如何製造不同的加密方法，則有三組的模式可以選擇，特定的模式所做的密文只能用該種模式來解，而 server 端要如何知道此封 email 適用何種模式？於是在 client 傳送時，會先傳一串代表等一下將要傳的檔案的訊息，這個訊息就包含了此檔案的加密模式。

以下的傳送流程圖的數字由小到大代表其處理的順序，圖下條列式說明前數字代表圖中編號的說明：

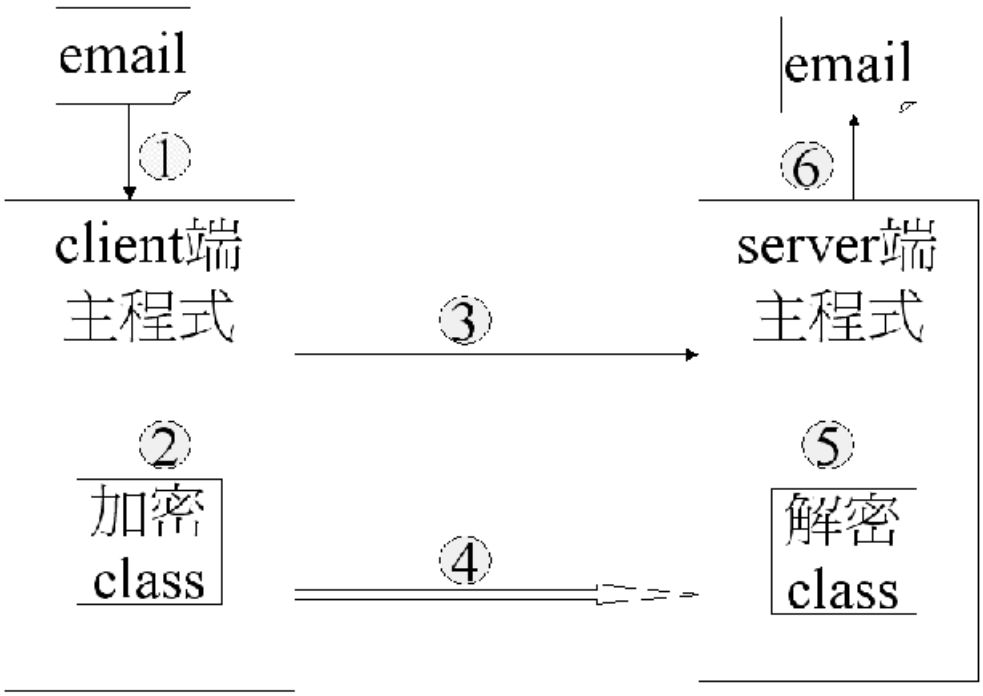


圖 4-17 加解密之間資料的傳輸流程圖

1.clinet 端主程式從 email 中讀取其中欄位，欄位值 0-4

2. clinet 端主程式傳欄位值當參數到所建構的加密 class 中，其中 0 代表不用加密，1-3 代表三種加密模式

3. clinet 端主程式先傳包含加密模式的資訊給 server 端

4. clinet 端主程式傳送檔案串流

5.server 端主程式接收之後，依所接收到的資訊處理所接收到的檔案

6.還原回解密后的檔案

4.6 壓縮及解壓縮

使用 Zip 壓縮管理檔(ZIP archive)，是一種典型的壓縮方式，而在 ZIP 檔案格式中，其尾端會有一個「目錄進入點」(dir-entry)，它是用來指出此 ZIP 檔案格式中，有多少個檔案，如下圖 4-18 所示

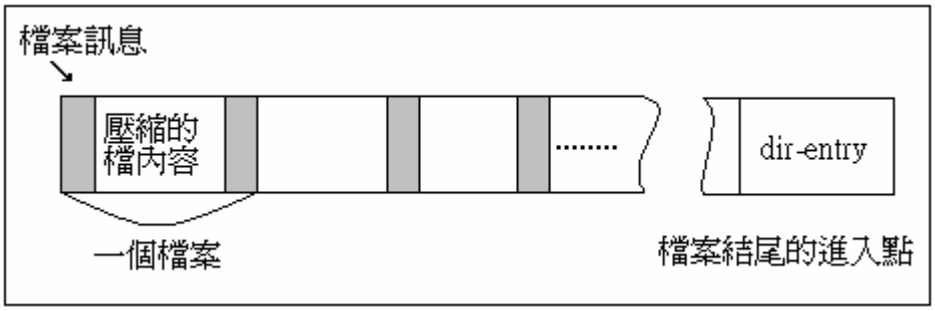


圖 4-18ZIP 檔案結構

在這裡我們稱 dir-entry 為 ZipEntry，他記錄的檔案開始的位置，我們知道 Zip 壓縮不只是對單一檔案的壓縮，他還可以對目錄、多目錄或多檔案的壓縮，而 ZipEntry 的重要性就在這裡，藉由它，我們可以找到該檔案開始的檔案訊息，有了檔案訊息，我們則可以從 ZIP 中解壓出我們要的檔案。

我們建立的 ZIP 格式就如圖 4-18 所示，要寫入時(壓縮)必須要先寫入到對應位置的「目錄進入點」，才能寫入(壓縮)檔案。而當要讀出(解壓縮)時，要先找到對應的檔案的「目錄進入點」(或檔案進入點)，然後知道該檔案在 ZIP 裡面的位置，才能開始讀這個檔案。

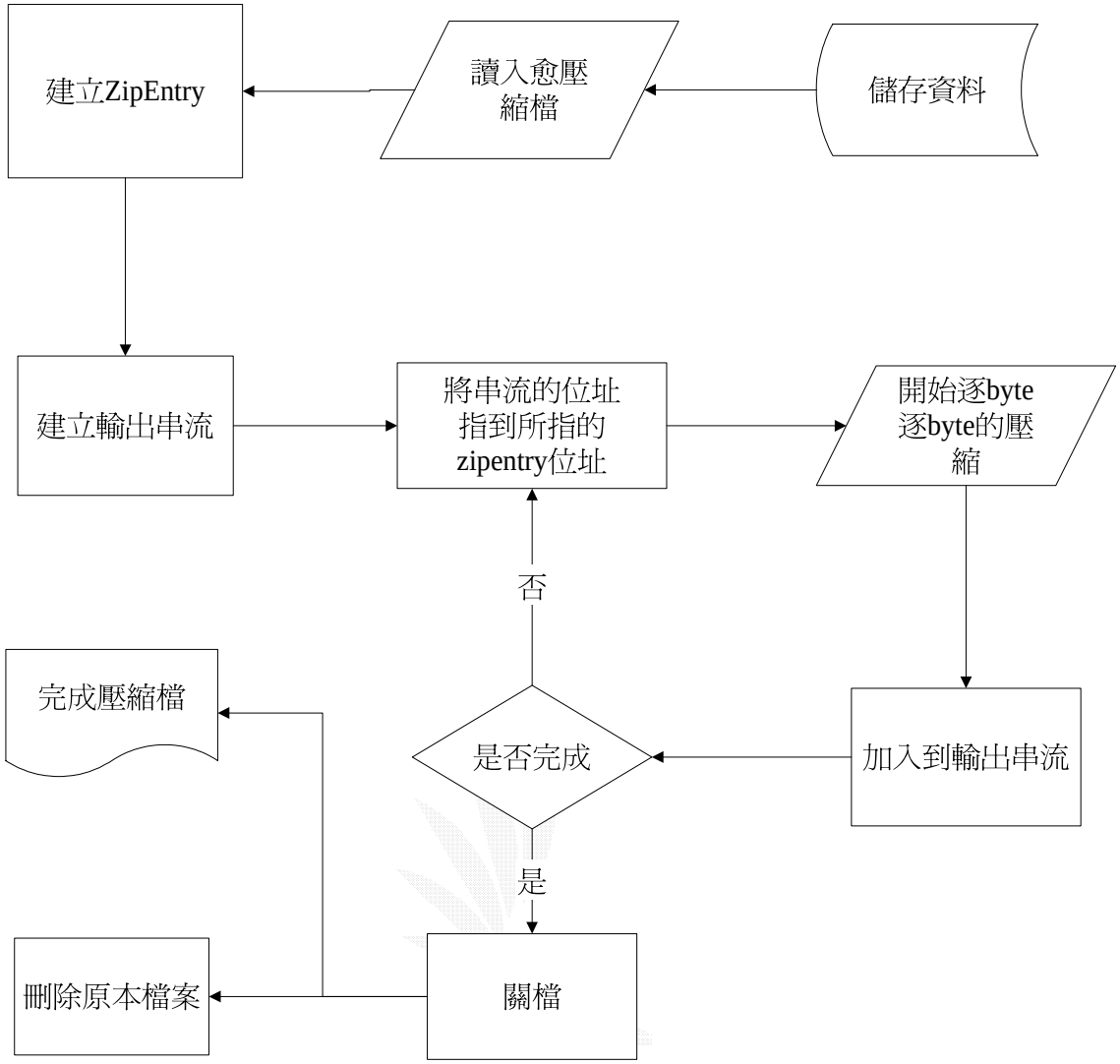


圖 4-19ZIP 壓縮流程

ZIP 內部的壓縮過程就如圖 4-19 所示，用下面的方法來輸入想壓縮的檔案。

```
FileInputStream input = new FileInputStream(輸入檔案);
BufferedInputStream bmyinput = new BufferedInputStream(input);
ZipEntry myzipentry = new ZipEntry(input);則是建立 ZipEntry。
```

解壓縮則跟壓縮相反，先建立欲解壓縮檔案的串流，再建立 ZipEntry，不論是壓縮或解壓縮都要建立 ZipEntry 的原因就在圖 4-18，因為他記錄著壓縮檔的位置，找到 ZipEntry 後，表示已經找到該檔案的開始，而由檔案開始的檔案訊息部分，可以得知檔案的長度，也就是該檔案的結束位置，有了開始和結束位置後，我們就可以將其讀出，經過解壓縮方法，還原為原來的檔案。

4.7 檔案的傳送與接收:

當 Client 做完該有的前置處理以後，在來就是針對這封信的寄出位置做傳送的動作。根據我們的需求，及系統分析的結果，我們利用 Java Socket 來達成送的目的。圖 4-7 就是 TCP/IP 的傳送基本概念，需要對方的 Hostname 或 IP number 在來是兩端的服務程式的 Port Number，最後是如何傳送檔案的 InputStream 或 OutputStream，有了這三個基本的元件，我們就可以組成我們所要的傳送需求。

因為我們需要有續傳的功能，以應付大型附加檔，如果遇到網路壅塞或對方 Server 目前忙碌中(因為 Server 端有最大接受連線的數目限制)，而必須等待，其次，我們還需要有可以在當目前這個郵件地址傳送失敗時可以選擇第二個或第三個可以取代的地址來傳送，所以我們建立一個 route.conf 裡面記錄著可以被取代的位址，在後面會說明。

傳送最基本的需求就是單一檔案的對傳，在圖 4-7 中我們可以很明顯的看到傳送的檔案必須用 InputStream 或 OutputStream 的形式來傳送，這沒問題，在傳送程式中我們直接用 FileInputStream 開檔

案，其形式就會是 Stream，然後用 Java Socket 的 getInputStream(接收) 和 getOutputStream(送出) 來達成，但是問題並沒有那麼簡單，根據系統需求，我們需要傳送一些基本的資訊，如檔名(用 Stream 的形式過去並不會傳送檔名)、有沒有壓縮、有沒有加密、用何種加密演算法、共有幾的分割檔案(續傳必須用到的分割檔判斷)，例如下面的範

例："21,0,4,name,13"

這一串資訊，內容依序分別是共 21 個分割檔，沒有壓縮、第四種加密方法，檔名為”¨nam” 目前正在傳送第 13 個分割。

因為我們必須傳送這串資訊，所以傳送就不單是 Stream 的問題了，所以我們用 DataInputStream(接收) 和 DataOutputStream(傳送)，這兩個方法將 InputStream 和 OutputStream 包成我們想要的格式，如 Integer(整數)、String(字串) 。

下圖為傳送資訊和接收資訊的意示圖：

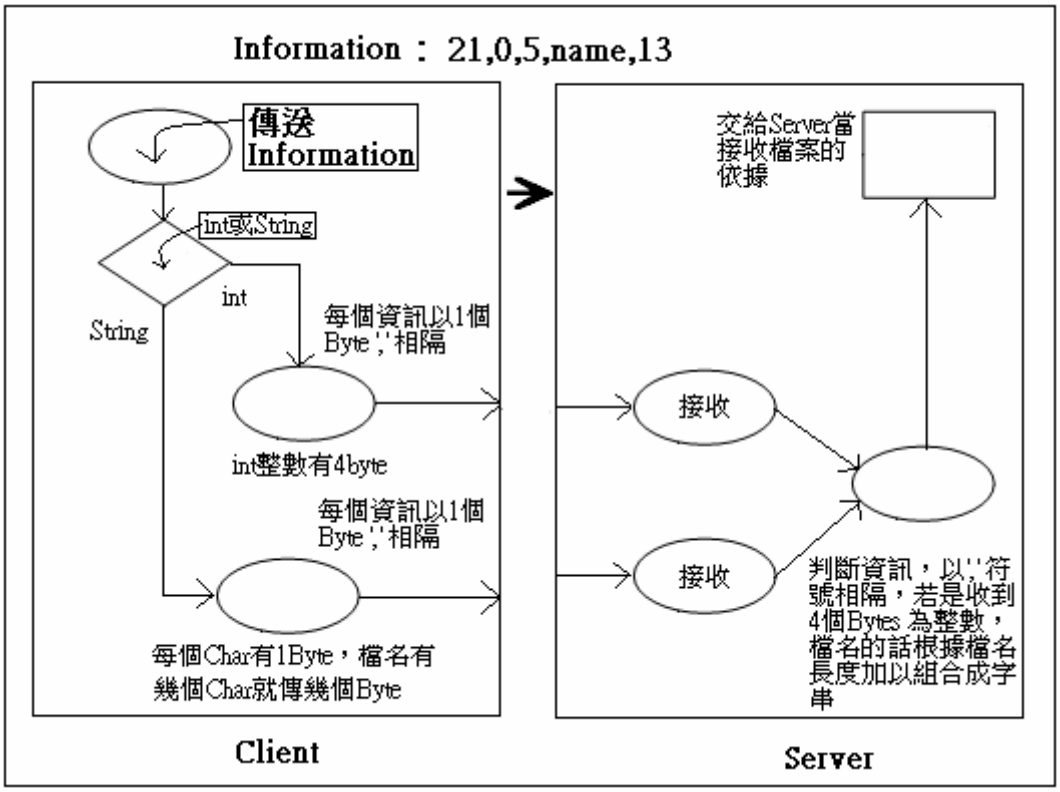


圖 4-20Client 和 Server 流程圖

圖 4-20，表示整個系統在傳遞資訊時的狀況，資訊(Information)為一串字串，Client 端會送出一段字串，裡面每個段落都會以`,`來隔開，只要 Client 端的送出方法和 Server 端的接收方法一致，如何接收檔案就沒有問題。

根據系統建立時的要求，程式必須有續傳的供能，以應付當附加檔案非常大的時候，網路的突發狀況，例如：網路突然中斷。因為檔案大，所以傳遞過程長，中間若是發生網路中斷，有了續傳功能可以節省一段傳重傳的時間。

續傳主要的功能是在 Client 端，Server 端在前面基本架構有提到，單純化的結果，Server 端只需要接收檔案，當接收完全部的分割檔後，加以組合，然後根據傳送前接收到的資訊，得知要不要組合、要不要解壓縮、要不要解密。

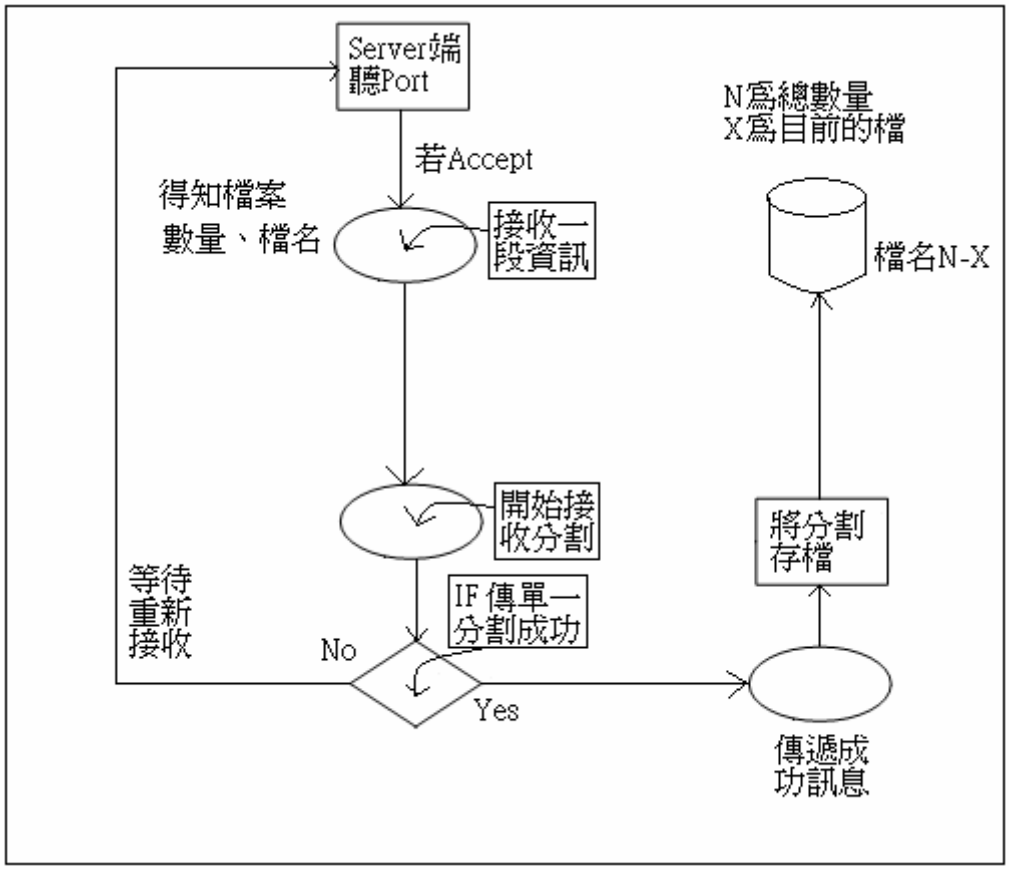


圖 4-21 傳送接收意示圖

4.7.1 Client 端的傳送和續傳方法

前面曾經提到，整個系統 Client 端和 Server 端的運作和互動主要是從 Client 端開始的，Server 端只是單純的接受要求，並根據 Client 端所傳的 information 一個字串裡面包含的資訊來進行動作。

假設現在有一封信已經到等待傳送的目錄位置，主程式會呼叫傳送程式(Clientsend.java 裡面的 function sending)並將在 XML 步驟所得到的訊息連同檔案一起傳給傳送程式，傳送程式接到這個要求以及這些資訊，會連線到目的位址，如果成功，傳送程式會依照設定檔內部的檔案分割大小將檔案加以分割，然後分段傳送(預設的分割大小為 5000 Bytes)，Server 端每接收到一個檔案，會傳遞一個數字回給 Client 端程式，如果一個檔案有 7 個分割，傳完第一個檔案時，Server 端會傳遞一個數字 1(整數為 4 Byte)代表第一個檔案接收到，當傳完第二檔案，Server 端會傳遞一個數字 2，代表收到第二個檔，餘下的以此種規則類推。

所以當 Client 端斷線後會將 Server 最近一次傳遞回來的數字，也就是目前晚完整傳完檔案的數在代號記下，並交給專門管理續傳的類別物件去傳送。

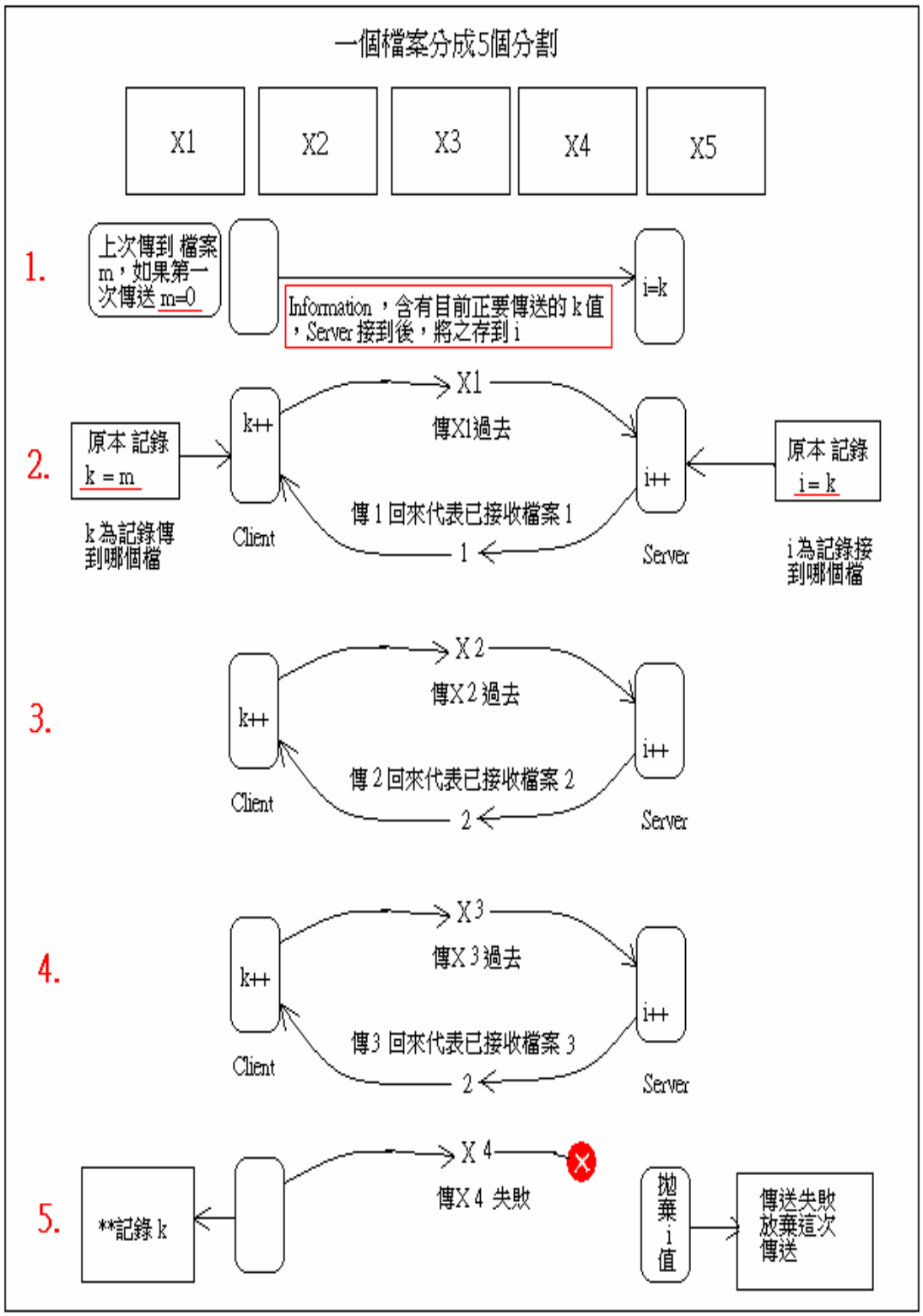


圖 4-22Client 端和 Server 端溝通圖

第一個步驟 Clinet 端的 m的意思是上次斷線後，傳到哪個完整的分割，將它放到 k，藉著第一個步驟傳遞 information 將 Server 端的 i 值存為 k,也就是從第 k 個檔案開始接收,第二,三,四步驟都是 Server 端接收完檔案將接收的檔案數傳遞回給 Client 端知道，而第五個不驟我們看到第 X4 分割在傳送中失敗，只要網路傳送狀態一發生異常，兩端都會馬上知道，Server 端這時會放棄這次工作，等待 Client 端是否繼續要求傳送；而 Clinet 端會將剛剛傳完成的檔案數 k 記錄起來，如果設定項目中的重複嘗試次數還沒到的話，則整個程序重頭，此時 k 會被當成 m繼續下一次的傳送。

4.7.2 Server 端的傳送和續傳方法

如圖 4-22 所示，Server 端在接收到 Client 端的要求後，會先接收一段 information，裡面記錄著整的完整的檔案共有幾個分割，和 Client 端目前要傳送那個檔案，只要完整有幾個檔案和目前重送完成的檔案數相等，就表示傳送完成了，有個傳送完成的這個條件，Server 端會開始後續的處理動作，如前面所述的解壓縮、解密、轉 XML 格式回 MIME 格式。

第五章遇到的問題及解決方法

整個系統看上去，功能看起來不多，讀取檔案、轉換 MIME 格式和 XML、加解密、壓縮、解壓縮到傳送看起來功能雖然不多，但是其實運作滿複雜的。

當我們看到這麼多功能，而且每個功能都可以獨立運作，最明顯的例子就是加密、解密、加縮和解壓縮，所以我們把整個系統分成可獨立運作的幾個部分，做成一個一個類別(Java Class)可以被建立實體並呼叫裡面的方法達成我們想要達成的目的，再來是以 Client 端和 Server 端各做一個主程式，可以統一管理整個 XML 流程從建立文件到傳送，再從接收到做一些 XML 文件處理再到目標目錄(Directory)等待使用者提取，我們將在下面兩節以 Client 和 Server 兩端的觀點來看整個系統建置時遇到的一些困難。

5.1 Clinet & Server:

Client 端需要的功能有：掃描目錄、轉 MIME 格式為 XML、讀取送出位址、讀取這個信封的處理分法、加密、壓縮、傳送、續傳等。壓縮沒什麼問題我們採用 ZIP 的方法，可以把純文字檔的信件壓縮。而 Server 端則需要結合檔案(分割檔) 、解密、解壓縮、轉換 XML 文件回 MIME 格式的郵件格式。所有的程式還有一項要點必須宗守，就是類別方法(Method)的參數和回傳值，必須影固定格式，這樣才能方便往後的擴充。

5.2 掃描目錄讀取檔案和信件優先權

當初掃描目錄只是將放信件的資料夾裡面所有的信件加以讀出做

處理而已，但是系統的要求裡面有一像是對信件做優先權的分類，而這個功能必須在這個階段加以完成，為了不要浪費開檔讀檔的時間，所以我們討論的結果，發現可以用檔名的排列當作優先權的判斷，就是說當我們掃描讀取信件的目錄時，可以將檔名以 ASCII 的字元順序排列，越小的排越前面，我們得到一個結論，例如：

檔案讀取順序：

0abc.eml > 9abc.eml > aabc.eml > zabc.eml

用上面這個準則，當掃描目錄類別內的方法(Method)在讀取檔案時，先以檔名來排列，因此，我們解決了這個問題。

5.3 XML 及郵件位址的讀取、信件處理方法任務判斷

XML 我們遇到的問題，首先是要何種方法建立 XML 文件。SAX (Simple API for XML)，是以循序的方法讀取 XML 文件，所以我們無法隨機存取 XML 文件，也無法任意的修改 XML 文件中的資料。另一個問題就是使用中無法得知目前正在此文件的什麼位置，因為 SAX 的循序特性讓它沒有跟元素的概念，所以我們無法任意的在文件中操作，一定要循序的方式，但是 SAX 卻有快速讀取的特性，也是因為 SAX 的循序特性。

DOM，是 W3C 制定的一套標準介面，在處理過程中，DOM 的解析器會將 XML 文件轉會成物件的模式(我們稱為 DOM 樹)，透過這個物件，我們可以任意的在 XML 文件中的任何位置做處理動作。我們都知道 XML 是分層樹狀結構的，DOM 處理 XML 的方式，就是將 XML 以根節點為樹根，往下同一層的為數中的同一層子節點，再往下用相同的方法建立子節點。

DOM 也有缺點，其將整個 XML 文件以 DOM 的樹狀結構的型式放在記憶體中，應用程式可以隨時對 DOM 樹中的任何一個節點進行存取與操作，也就是說透過 DOM 樹，應用程式可以對對 XML 進行隨機存取，這種方式為應用程式的開發帶來了很大的方便性與靈活性，但是，DOM 把整個 XML 文件以 DOM 樹的型式都塞進記憶體中，所以當 XML 文件較大時，或節點結構較為複雜時，對記憶體的存取需求也相對的較高，所以執行效率並不十分理想，我們這組討論時也發現這個問題的所在，但是因為 DOM 可以輕易的對文件隨機操作，而我們整個系統中，讀取郵件寄出位址，以及對信件該用什麼方法處理(任務)，卻可以輕易的用 DOM 來達成，所以我們決定採用 DOM，來做為建立 XML 文件的方法，而信件中，如果附加檔太大的話，我們用另一種處理方法，就是分兩階段建立 XML 文件，第一次建立 XML 文件至，郵件內文前，第二次專門對郵件的內文及附加檔做處裡，使的整體效率不會因為內文和附加檔太大而影響整體的效能。

最後，因為我們採用的是 DOM 的方法，所以讀取 XML 郵件內容的郵件位址問題有因此得到解決。

5.4 加解密

在不同 Java 版本遇到的問題，JDK1.4 版具有最新的 API 以及方法(Method)JDK1.3 版本沒有內建加密要用到的 JCE，要另外抓取 JCE 的套件，裝在 Java 的預設 Classpath 就好了。

5.5 傳送和續傳

這個部分為整個系統最為棘手的部分，普通的傳送還好，只要建立好 Client 端和 Server 端的 TCP/IP Socket 連線，問題及得到解決，但是系統要求中需要有續傳的功能，定義為「當傳送一信件中，如果遇到網路突發狀況，如斷線或其中一端當機，該有的即時處理」。我們

先想要把問題單純化。首先，續傳在 Client 端和 Server 端兩端分別需要哪些工作? 可不可以將所有工作固定交由一方達成，因為如果和 Server 兩端都同時提出要求，將會導致整個程序的混亂，於是，我們想模仿一般 Server 模式，即是以 Server 提供服務，Client 端的問題交由 Server 處理，但是這樣不但不能將問題簡單化，反而增加兩端的設計困難度，檔案的處理已經是一個困難點了，如果再讓兩端的問題複雜的化，整個程式的行為將會非常複雜，原因在傳送信件的定義是將信件由 Client 傳給對方 Server 端，而且又是續傳，需要將檔案分割，如此一來，Server 將無法正確的傳來的片段檔案中得知 Client 的情況,如果要在接收時採用「information」一一的去記錄每個 Client 端的要求，更是會造成 Server 端的負擔，交給 Client 端處理的話，因為 Client 端掌握全部的信件，可以隨時提出傳送要求，而我們不要 Server 端在斷線後去一一的記錄位成功的傳送資訊,所以 Client 端在每次傳送前，都要傳送一段「information」,裡面記錄著希望 Server 端該用什麼方法處理信件，也包含了總分割數和正在傳送那個分割的資訊，因此，Server 端的工作得到單純化。

Client 端主控整個傳送並不是完全就沒有缺陷了，一封信有固定的可設定傳送次數，和中間相隔時間，如果這兩個數字都很大，或其中一個相當的大，那整個 Client 端的程式運作都會被停住，全部都等待這個信件的傳送結束，當我們發現此問題，再來就是尋找解決的方法。我們將續傳中用到等待，就是屍拜傳送時間，以及第一次以後的傳送動作都寫在另一個類別裡面，而讓這了類別繼承 Thread(執行緒)，讓它獨立在住執行緒外(Main Thread)，這樣一來，後面的工作不需要等待前面信件的續傳過程。

第六章未來的發展與心得

6.1 系統未來展望

目前電子郵件的寄件格式都是用 MIME 格式配合上 SMTP/POP3 來傳送與接收，而且適用於普遍的用途的，我們建立這個系統的目的，就是想直接將 XML 的好處用於郵件的傳輸上面，因此為了一個機構、公司或團體設計一個可以及時傳送的、可以具安全性及隱密性的、可以容許網路或主機錯誤的系統，再配合上目前的 XML 技術，構成一個可以在一個公司的總公司、各地分公司或是組織在各地的分部之間可以傳遞訊息的系統。

XML，雖然是為了能在網路環境下有效發展而設計的，但 XML 也可以在網路以外的環境運作，例如和資料庫的合作、商業上的應用、及資料交換場合。所以我們採用 XML 當成資料傳遞的內容的意義，就是在他的多用性、以通用性。只要根據同一個 XML 規則定出來的所有 XML 文件，都可以被想要利用供做輕易使用。

隨著網際網路的普及，各企業或組織單位都有了 E化的網路系統，和各地之間越來越頻繁的通訊及龐大的資料交換，我們可以想像，並不是每家公司或每個單位都用同一種資訊交換系統，例如：微軟可能用 SQL SERVER、IBM 可能用 DB2 而 Oracle 用 oracle 等等，而假設我們想要轉換各種格式做資料的轉換，過去是使用 EDI(電子數據交換 Electronic Data Interchange)，EDI 將資料格式化後經網際網路做傳送處理。EDI 可以減少資料處理的費用及複雜度，但在實際上卻有困難，因為它嚴重缺乏可延伸性，如果要修改，成本很高，工程也浩大。

但是現在我們可以利用 XML 的通用的優越性，來定義各種資料，屆時我們可能採用企業或組織內部之間用 XML 做資料傳遞交換，或者是企業組織和別的企業組織之間的訊息傳遞交換。再加上如果該企業組織有使用資料庫或是列印報表、將資料公佈於網頁、各種型式的資料處理，如果能藉由 XML 統一格式的處理，及隨時具備延伸擴性的性質，成本和處理複雜度都會相對的降低。

XML 最主要的功能，在於保持使用者介面與結構化資料的獨立，例如在 HTML 中我們會用標籤來告訴瀏覽器資料如何顯示，但是在 XML 中，卻可以利用獨立的 XSL 和 CSS 定義資料內容，XML 可以將資料的內容及如何處理分開，可以適用於更多的場合。

XML 為網路應用程式帶來下列優點：

1. 資料搜尋變的簡單，藉由 XML 一致的方法，在跨網站搜尋的過程會變的容易。
2. 可以整合不同的資料來源，XML 可以將，不同來源的資料加以整合，應用程式可以更輕易的處理資料。
3. XML 藉由 DOM 的解析，可以將獲得的 XML 文件做各種不同的用途及運算，並不是整個文件只有單一功能、目的。
4. 可以局部的更新，XML 可以指更新文件中的部分資料，而不必每次要跟新文件時都要重傳整份文件，只需要傳送部份更新的資料即可。

整個 XML 郵件傳送系統再加上可以加密的功能，使的我們也有可以傳送機密資料的選擇。而且只要符合主程式的呼叫，可以隨時擴充新的加密演算法。

6.2 心得：

心得(陳威宇)：

這次的專題我的部分是加解密及壓縮解壓縮的研究與實作，以及配合 eml 轉 xml 的方法，還有 java 介面的製作(雖然醜醜的)。關於壓縮與解壓縮，我想現在的坊間有各種不同的壓縮軟體，連 windowsXP 甚至把壓縮與解壓縮變成像放到資料夾與從資料夾拿出來這麼的方便實用。其實用 java 寫壓縮與解壓縮的程式並不難，掌握 zipentry 的連貫性就可以解決。不過直得一提的就是關於壓縮方面，一些細節的注意可以使壓縮速度變得更快，例如加緩衝、處理 io 串流等。

用 java 來寫加解密的程式應該比其他語言好寫的多，不過常常碰到的狀況是，我寫的出來的 java 程式 compiler 並不看的懂，或是我的電腦能跑但別人的不行，或是 compiler 過了而且沒問題但是偏偏 run 時才出現一大堆的例外處理，用成執行緒來呼叫時連跑到例外處理都不會只知道程式一動也不動。明明都合乎文法，原因就是出在 java 中的 jce。1.4 版已經把 jce 融入，但 jdk1.3.1 卻沒有，除此之外，還需安裝 jre(java 執行環境) 才能安裝 api，安裝了並不能用，還需要手動驅動 java.security 檔，而這些問題是一頁一頁瀏覽 java 的英文網頁才獲得解答。加解密程式碼要比壓縮的還要難理解的多，參考了多個範例和網頁的教學後竟然也實做了多種加密模式以供選擇。

還有一個部分就是搭配陳弘奇所轉譯成 xml 的 eml 文件，再把他還原回去。這是最早開始的著手的部分，也是做最久的部分，因為一值在思考其正確卻又要提高效率的問題。而我這個部分是用 java 的 jsax 來作。

總而言之，這次的專題讓我的 java 從零到有，還有 java 的各種 io 的做法，介面類別設計與驅動，以及提高程式設計及除錯的能力，還有享受和大家一起努力做專題感覺的樂趣。

心得(黃政勳)：

這一次很高興能參加此專題的研究，我的研究範圍在於 Java 網路傳輸實做和續傳、XML 方面的概念研究，這研究讓我擁有感受很深的專業知識成長。

在於整組在於 E-mail 轉 XML 的研究時，第一次接觸 XML，實在是一件令人很頭痛的事，因為 XML 還是一各不是很通流的語言，它到目前都尚未成熟，因此都常常去找學長研討，才漸漸的對於 XML 的 DOM、SAX…等，有了概念之後，開始選擇語言去撰寫時，大家決定用 JAVA 去撰寫 XML 程式，開始研究 JAVA 和 XML 之間的關係，光是一開始的開發系統的建造，也就是安裝 JAXP API 的套件，我就遇到設定的困難，還好我的組員都能指導我，一起研究，還慢慢的開發出 XML 程式。

對於 JAVA 的網路傳輸時做方面，對我來說是一個對專也能力成長對大的地方。雖然課堂上接觸過 JAVA，以為自己很了解，到了接觸專題後發覺，它的功能真的很大。一開始時實做時，要先做一個即時掃描的的程式，它必須隨時的偵測檔案，並且傳送檔案到別的資料夾，聽起來如此簡單的程式，卻是跨越 JAVA 的資料串流概念和多執行緒概念，這都是我以前沒學過的，尤其是多執行緒，我是利用繼承 Thread 類別來實作，常常遇到與 JAVA 的例外處理衝突，讓我常常需繞道而行，想出其他的概念解決，JAVA 原來用起來也是很多的不懂地方。

JAVA 的網路傳輸方面，是利用 JAVA ServerSocket 類別的程式去構造的，這也是我第一次接觸，如何利用 Client 和 Server 之間的溝通去連線傳輸，比較難就是續傳的概念，因為網路的中斷有太多種的情況，而每一種的狀況都要考慮，難而最難的就是如何讓 Client 和 Server 之間溝通去判斷是否出現狀況，而狀況產生是如何解決，是最難的。組員的協調之後，決定利用檔案分割一定大小之後，讓 Client 和 Server 之間能夠溝通傳送數目，利用這樣的方法來判斷中斷後的事後處理。對於優先權方面也是有考量的，也就是會對網路的連線會先測試，如果不行的話，才會有第二優先權的選擇，以確保網路的連線和傳輸的品質。

最後我的心得就是，用專業方面的佼不來說，我的 JAVA 網路方面和檔案的方面知識，增進了非常多;而整個專題來說，感謝組員的氣氛帶領，讓大家都很熱心的去完成，具有了團隊分工的精神，我想這是對我影響很大的關鍵。

心得(陳弘奇)

這個專題的程式實作，我們採用 Java，之前我慣用的程式語言是 C和 C++，而且就算是用 C++這個物件導向的語言，以前在實際寫程式上，也很少用到物件導向的關念去寫程式，甚至，以前都沒有碰過 XML 相關的東西，為了做好這個專題，還特地買了 6 本 Java 和 XML 相關的書籍，現在做完以後，我發現自己除了有了整個程式以團隊的方式來開發的經驗以外，自己還徹底的學會物件導向的程式寫作和 Java 語言的熟晰程度，更是了解了 XML 相關的支援 API 和其用法，獲益不可謂不多。

整個專題，開始到現在，大概歷經 15 個月，前面 6 個月不斷的和實驗室的學長和同組的同學討論整個程式的架構，花了好一段時間，在 8 月多，除了續傳外，整個程式的架構都完成了，包括一些小區塊等，期間也有幾次測試，我要感謝同組的好伙伴陳威宇，常常熬夜一起寫程式，尤其是 Client 端和 Server 端兩端程式，只要整個架構一變動，兩個主程式都得跟著改，而且這經過很多次的改版，才有今天的風貌。本來每個程式區塊之間都互相獨立，設計時的考慮和設計完有落差，使得初版的主程式很混亂，一個小地方改，整個主程式都得跟著改，那時候除錯非常不易，後來才忽然警覺到每個區塊和主程式的控制都要有一定的規格，改正了系統分析的結果，終於確定目前的架構。

最後，就是續傳這個程式了，本來我們的想法很複雜，傳送、接收兩端都想要有所動作，傳送能測試接收端有沒有反應，接收端會一直等待傳送端。後我發現這樣的結構雖然可行，但是會讓兩端的程式都非常複雜，所以才會有傳送端先送一段小資訊(Information)，用這樣的方法，Server 端只要讀到傳送端先傳的一段資訊，Server 端只要解讀到 Client 端的要求，一切照做就可以了，兩端都很輕鬆，也減少了程式撰寫時的負擔。

參考資料

[1]江義華，java完美經典，p8-3~P15-79，金禾資訊，mar.2003

[2]蕭明城/ 周岱琳，java I/O與通訊介面，金禾資訊，sep.2002

[3]Shelly/Cashman，系統分析與設計，東華書局，aug.2001

[4]陳會安,XML網頁製作徹底研究,pp.112-135, 旗標股份有限公司,May.2002

[5] 陳會安,JAVA2 程式設計,pp.182-205, 學貫行銷股份有限公司,Mar.2002