逢　甲　大　學

資訊工程學系專題報告

具遠端環境安檢機制的分散式加密通道
-Mobile Agent 的應用

學生：吳國銓(四甲)

陳重宏(四甲)

指導教授 ： 李維斌 老師

中華民國 九十二年十二月

目　錄

圖目錄

摘　要

行動代理人（Mobile Agent）是一種能夠同時一邊在各電腦終端間移動，一邊能夠自主地（autonomous）分散運算處理的一種程式。它具有提高分散計算系統效率性及便利性，同時又可降低分散式系統上的傳遞延遲與電腦間傳送次數等優點，因此可以說是在分散式處理系統中相當受到矚目的。

我們的專題主要是使用 Mobile Agent 的技術來實作出加密傳輸,因為現今的加密傳輸技術大多還有其缺點存在,而利用 Mobile Agent 的自主性及在分散式系統的效率性,來加強加密傳輸的功能,增加傳輸的安全性。

第一章　研究動機及目標

1.1.研究動機：

一般房屋業的業務員在各地執行業務工作，會透過行動裝置(如筆記型電腦)來與公司內部的資料庫做資料上的交換，記錄客戶、業務等資料，在這樣的分散式系統下，假若業務員的主機安全防護不足而被入侵，其與公司所建立的連線則會對內部造成危險，若沒有使用好的安全加密機制，無疑就好像將其內部的環境暴露在外，因此，如果能夠先確認業務員的環境安全之後才建立連線的話，就可以防止連線被惡意的入侵。除此之外，業務員也不須安裝任何的程式，只要向公司的專屬通道要求之後，再由公司端配送程式即可，如此一來，不僅行動便利，也更能增加安全性。

1.2.研究目標：

發展一套具遠端安檢機制的分散式安全加密通道系統，其能夠針對使用者端進行防火牆及其防毒軟體之類的安全性檢查的系統。

1.2.1 分析：

為達成目標，我們將主要的問題列舉如下：

1. 如何做到遠端的安檢？
2. 如果遠端的安檢不通過，是不是能夠要求其更新環境以增加安全性？
3. 如何做到將程式配送到遠端而非要求遠端事先安裝程式？
4. 如何建立遠端與內部的安全通道？

根據以上的問題，我們發現到 Mobile Agent 可以幫我們解決以上的問題。

1.2.2 Mobile Agent 的特性

為什麼 Mobile Agent 可以幫我們解決問題呢？

1. Agent 是一個獨立的機制，它具有能夠掌控自己的行為和內部狀態，而且不受其他系統和使用者之干涉，可獨立自我管理、控制。藉此，公司內部可配送獨立的 Agent 至遠端做一些處理(如安全性檢查)，並且回報遠端的安全等資訊 。
2. 公司內部的 Agent 可以與配送出去的遠端 Agent 交談，甚至控制其行為，藉此，倘若遠端的環境安檢不通過的話，內部的 Agent 可控制遠端的 Agent 將其銷毀(dispose)。

3. Mobile Agent 有特定的 Runtime System，Agent 可在裝有 Run-time System 的環境下活動，遠端只要有此環境，便不用安裝任何程式，只要從公司內部配送執行即可。

第二章 Mobile Agent

2.1 什麼是 Mobile Agent

所謂的行動代理人(Mobile Agent),其實也是代理人技術(Agent technology)的一種。其基本原理就是當 Agent 接收到終端使用者之請求命令,便即時地處理使用者所交付之任務,並依實際之情況可從網路一端 host 移動至其他另一端 host 處理,最後並將結果回傳給使用者。而實際上所移動的對象不僅是本身的程式碼和處理終了結果,其中還包括了程式執行狀態,也就是程式內的變數,甚至是整個物件。另外,在終端間移動中的 Mobile Agent 即使是在無法和使用者相互通信的環境上,也能夠將移動前的程式憑藉自主地的分析、判斷並繼續執行處理;在此所提到的自主性(Autonomy),是指 Mobile Agent 具有掌控自己的行為和內部狀,且不受其他系統和使用者之干涉,乃能獨立自我管理、控制,也就是說能夠從自己本身的知識或所收集到的情報來加以分析判斷,最後並自行解決問題的一種機制。其一般處理方式是透過事先安裝在電腦上之 Run Time System 或是藉由稱作 Agent host 的軟體來實現的;與以往代理人技術不同的是,Mobile Agent 更具備了自主性(Autonomy)與機動性(mobility)。

2.2. Aglets 介紹

Aglets 是由 IBM 東京基礎研究所發展出的一種 Mobile Agent 系統，其不僅是代理人，還包括系統本身，全都是由 Java 語言所開發完成的，所以系統具有相當高的移植性，Aglets 包含了 Aglets API 的 packages、documentation、sample Aglets 和 Tahiti Aglets Server(圖 2.1).Tahiti 是一個 java application，它允許使用者接收、管理和傳送 Aglets 給其他有 Tahiti 的電腦。

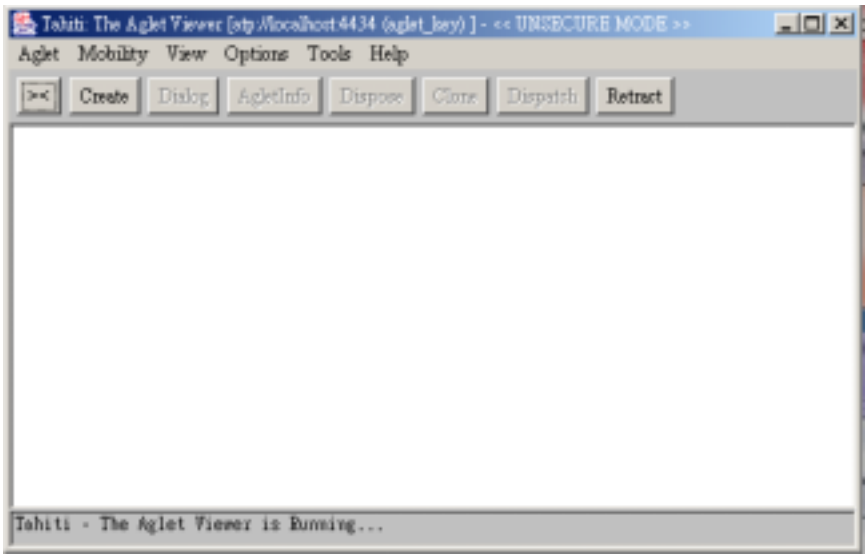

圖 2.1 Tahiti Server

Aglets 基本的元素：

Aglets 的基本元素包含了 Aglet、Proxy、Context 和 Identifier

- Aglet ： Aglet 其實就是有代理人環境的 Host，它是自主的，因為它是在到達 host 後，執行在自己的 Thread 中。

- Proxy ： Proxy 就是 Aglet 中的代理人，保護 Aglet 避免直接執行 public methods。圖 2.2 表示 Aglet 和 proxy 之間的關係。

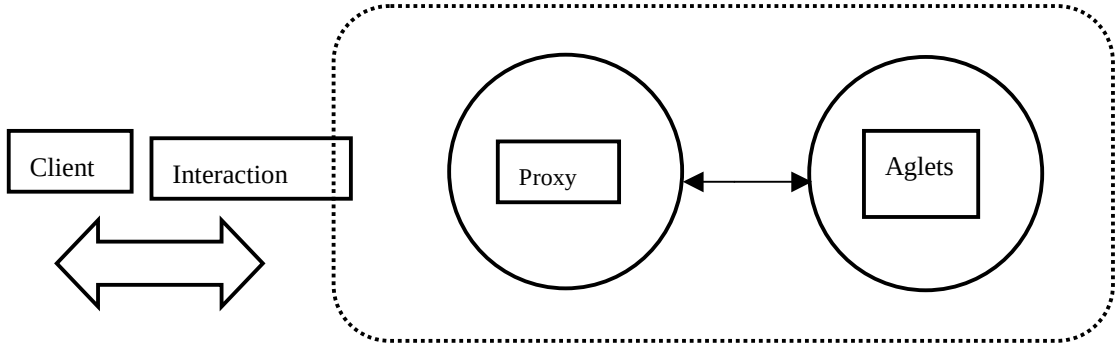

圖 2.2 Relationship between Aglets and proxy

- Context： Context 是 Aglet 的工作區，它是一個固定的 object 負責提供維護和管理執行中的 Aglet 的工具，並防止有惡意的 Aglet 進入。
- Identifier： Identifier 是用來界定每個 Aglet，每個 Aglet 都是 unique 而且有其一定的 lifetime。

Aglet 基本操作元件包括了下面四項：

1. Creation： Creation 中包含了 Create 和 Clone， Create 就是產生一個 Aglet，新的 Aglet 被指派一個 ID， Clone 則是把一個 Aglet 幾乎完全相同的再複製一份。
2. Disposal：當 Aglets 執行完其任務，或者要結束 Aglet 時，便會使用到 Disposal 元件，來將 Aglet 丟棄。

3. Mobility: Mobility 中包含了 dispatch(派遣)和 retract(返回)元件，dispatch 即是將 Aglet 從一個 context 移動到另一個目標 context 中，在那重新執行;而 retract 則是從目的端 context 回到原來的 context。

4. Storage： Storage 包含 Deactivate(暫停)和 Activate(啓動)，Deactivate 主要是讓 Aglet 暫停執行並將其儲存起來;Activate 則是讓 Aglet 回復，重新執行。

圖 2.3 表示 Aglets components 之間的互動及 Aglets 的生命周期。

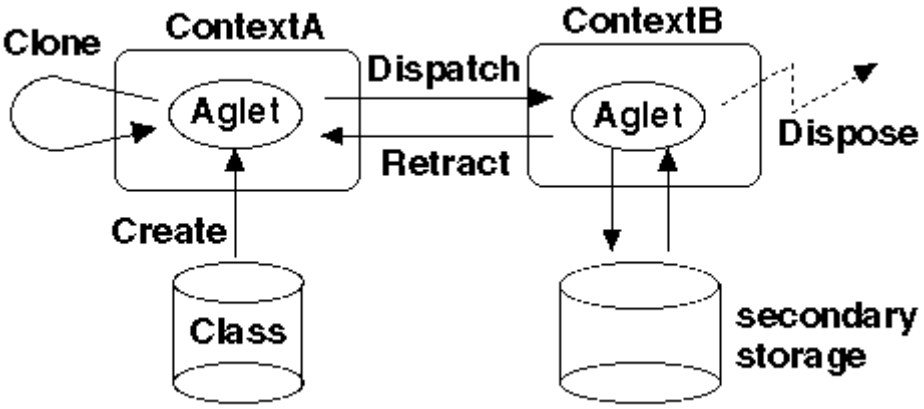

圖 2.3 Aglets Life-Cycle Model

2.3.為何使用 Mobile Agent

因為我們的目標是實作出一個在分散式系統下的安全加密通道，然而 Mobile Agent 在分散式的系統下有以下幾個優點:

1.遠端分散式的處理:

在無線網路的環境裡，由於通信延遲較大造成分散式系統的性能上之瓶頸（bottleneck），在此，如果能夠藉由 Mobile Agent，必能降低通信成本。遠端處理就是以 Mobile Agent 將執行中的分散處理程式傳送到其他電腦繼續執行，並在傳送端 host 傳送程式碼前先行壓縮，可使其傳輸通信量減至最小限度，以分散各電腦及網路的負擔。另外，如通信傳輸安全許可，則 Mobile Agent 可藉由自己本身程式來直接控制，並存取其他電腦上的運算執行資源，大為提昇其效率及靈活性。

2. 非同步處理:

再就分散式系統觀點來看，在傳統主從架構（Client-Server Paradigm）中運作時，Client 端需要一直與 Server 端保持連線，否則無法正常運作，造成網路頻寬浪費；而 Mobile Agent 技術則可解決這個問題，當傳送端 host 將程式送到接收端 host 時，傳送與接收兩端就不需再保持連線通信，Mobile Agent 即可獨自在接收端 host 上執行，大幅降低網路流量。

3. **分散程式設計的簡化:**

過去分散式系統,例如在主從模式下有 Client 和 Server 兩種的程式，當遠端程序呼叫（Remote Procedure Call， RPC）時，會另外產生 stub 和 skeleton，且 Client 和 Server 兩程式必須事先安裝在遠端電腦上，相當浪費資源；而在 Mobile Agent 方面，只需要在各電腦裝置上安裝 run time system，各電腦便具有移動和自主的處理能力。run time system 提供如下兩種主要機制：

A. Mobile Agent 於各電腦間的移動：

實現 Mobile Agent 的最大特性－移動性。

B. Mobile Agent 之執行管理機制：

負責管理移動後的代理人之運作,也可藉由此一機制來使安全性受到保證。如果網路上的各電腦有了這系統，則利用 Mobile Agent，程式得以事前配置。另外，程式主體本身也不須在各電腦間傳遞，此也可以藉由代理人的分散移動而取代。因此，我們可以僅透過 Mobile Agent，不需考慮網路架構即得以實現分散程式設計（distributed programming)；即使是通信失敗等例外處理程序也得以簡單化。在過去也有許多程式在電腦間移動之相關技術，例如 Java Applet 或 Postscript 等，都以遠端電腦處理為前提，但是因為傳送對象不包含執行中的狀態，故接收端 host 需從程式最初始狀態重新執行；相

反地，Mobile Agent 技術可同時傳送程式碼與執行狀態，其中包括變數、程序及物件之傳送。因此，Mobile Agent 技術可以說比原有之技術有更長足的進步。

然而為什麼我們會選擇使用 Aglets 呢?在我們專題研究過程中，我們比較過幾種不同的 Mobile Agent 系統，如 General Magic 的 Agent Tcl、IBM 的 Aglets、及 ObjectSpace 的 Voyager 等等，我們決定使用由 Java 語言所寫成的 Aglets 來當我們的系統工具，其中原因有以下幾項：

1. 在跨平台上：Java 是一個新一代的語言，其物件導向的語言設計，以及所強調的『Write once，Run everywhere』，更是符合我們專題的需求。
2. 在設計上：IBM 的 Aglets 本身的系統具有相當完備的設計，而且文件更是齊全。Aglets 對一個 Agent 的行動，如 Dispatch、Dispose 等等，都有相當程度的實現，跟我們的需求非常符合。

所以我們選用了 Aglets 來實作我們的專題實作的平台。

第三章　系統介紹

3.1. 系統需求及安裝

軟體需求：

Windows2000/Windows XP　以上的OS
Java SDK
The Aglets Software Development Kit

硬體需求：

CPU：基本上可以正常執行Windows2000 & Windows XP 即可。
RAM：建議128MB 以上或256MB。
Network

安裝：

首先到日本 IBM 的網站上下載 Aglets

(http：//www.trl.ibm.com/Aglets/relnotes11b1.html)

將 Aglets 解壓縮放到目錄中

將 Aglets1.1.0\bin 底下的 Agletsd.bat 中的環境變數設定好

1. set AGLETS_HOME=(Aglets 目錄位置)
2. set JDK_HOME=(JDK 的目錄位置)

設定後即可執行，輸入基本資料就可進入

可設定 port：

```
c：\Agletsl.1.0\bin>Agletsd －port  9999
```

3.2. 系統整體架構

我們的專題大概可以分為三個目標:

1. Client 端安全檢查：

因為安檢程式不能讓 Client 端使用者任意更改，因此我們將程式碼由 Server 端以 Agent 傳送到 Client 端的執行環境中執行，因為 Agent 有其自主性，所以不會受到系統或使用者的干涉，能夠達成其安檢的目的。

2. 安全通道：

安全通道的建立，我們採用的是 I/O 串流來將 Client 端和 Server 端接起來，Client 端檔案的輸入串流會先接上密文串流，在密文串流中作加密的動作完，再送到網路串流，由 Agent 送到 Server 端，而 Server 端則是由網路串流接收後，送到解密串流作解密之後，再接到輸出串流，以完成整個資料的傳送。

3. 整體架構：

這個部分主要是 Agent 間的交談，可由圖 3.1-1 及 3.1-2 中看出 Agent 間的訊息傳遞。

（非點對點模式）

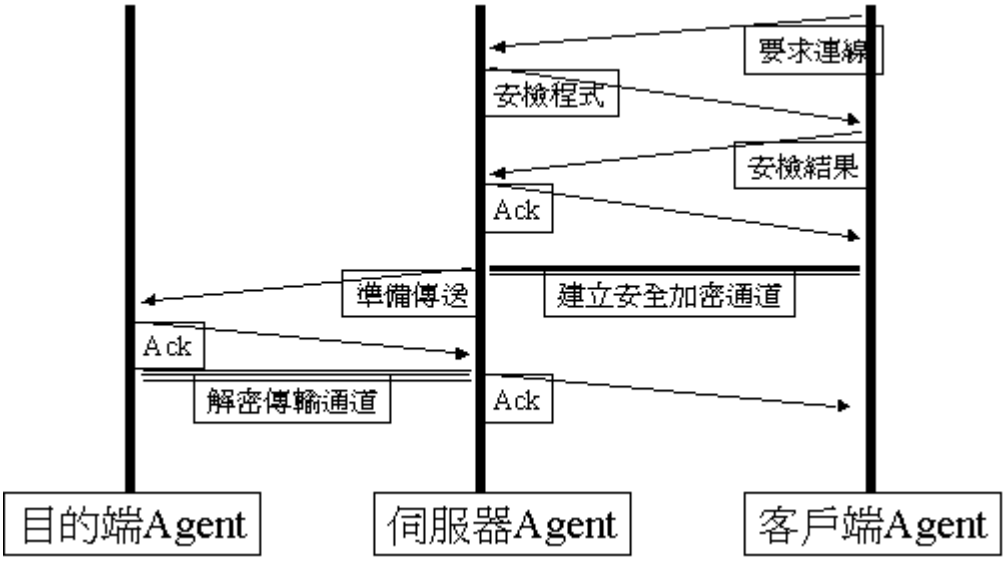

圖 3.1-1　交談架構圖(一)

（點對點模式）

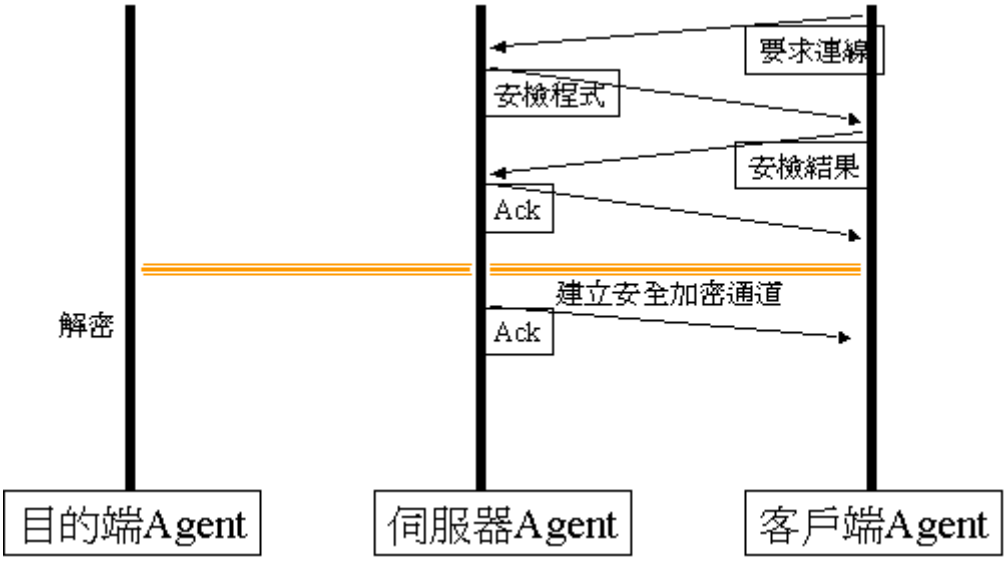

圖 3.1-2　交談架構圖(二)

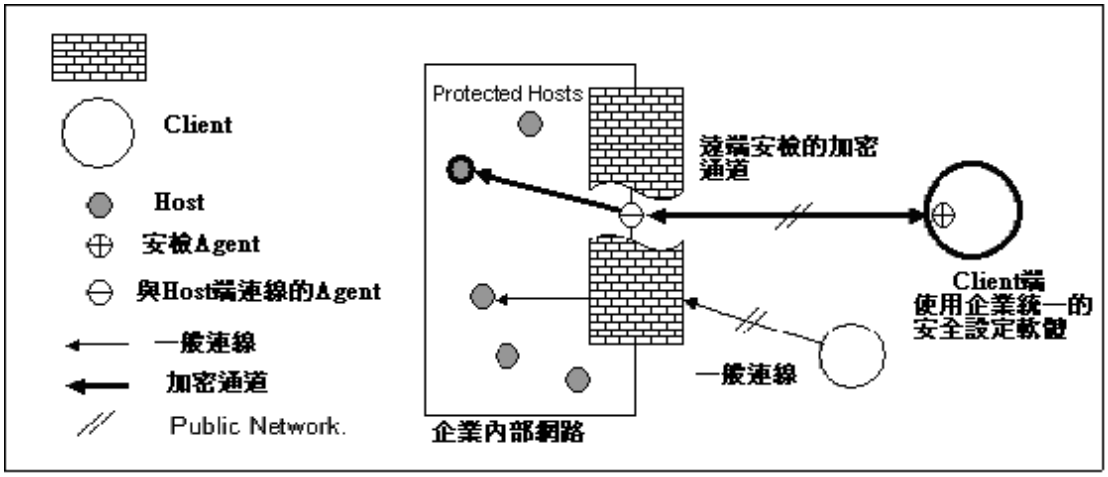

圖 3.2 Agent 傳輸意示圖

我們的專題大致上可用這幾張圖來加以說明。

設備：

1. Client 端
2. Server 端(Firewall)
3. Host(Destination).

流程：

1. Client 向 Server 要求安全加密通道，Server 便產生一個 Agent 送至 Client 端。
2. 此時 Agent 會先在 Client 端檢查其環境的安全性，如：防毒軟體版本、病毒碼、防火牆等等。檢查通過之後，才正式開始建立與 Server 的連線。

3. Client 利用 Agent 內部包含的 key(此為產生 Agent 時的 Agent ID)來與 Server 建立安全加密通道。
4. Client 透過此通道加密封包資料送至 Server，使用者可選擇讓 Server 作解密處理再轉送封包資料至 Host，或者是不經處理直接轉送至 Host 再行解密。

3.3. 目的端主機

目的端主機也就是 Host 端，資料傳輸的最終目的，可視為區域內部網站中的一部主機，在平常的時候只會處理內部的資料傳遞、或者向外傳送資料，並沒有任何的 Agent 常駐來處理封包，此端主要的功能是接收來自防火牆所傳送過來的 Agent，然後這些 Agent 是通過防火牆的認可，都算是安全的 Agent。

3.4. 來源端主機

在這裡，使用者可向 Server 做要求，主要是傳輸的來源。

3.5. 防火牆

因為市面上的防火牆種類很多，而各家的技術又有所不同，所以我們只作出一個具有封包檢查的機制來接收和傳送 Agent，並沒有包

含完整的防火牆在內。而此端會接收 Client 送過來連線要求的 agent，然後會發送一個 Agent 到 Client 檢查 Client 的安全機制，若檢查結果正常，便允許 Client 端傳送資料，若不通過則中斷其連線。

在這裡負責接受加密要求、監聽封包、傳送 Agent、過濾封包等工作。所有的 Agent 皆由此送出，其他端的主機並不直接處理 Agent，而是等待接收 Agent 之後，才能夠處理封包的加、解密等等。

1. 主要工作：

有一個 Agent 負責常駐，專職監聽某個通訊埠上是否有遠端要求建立安全的加密通道，若有，則送出 Client 端的專屬 Agent 來負責封包的加密、傳遞。接著，在遠端的使用者便可利用此 Agent 來傳送並且加密傳遞資料，然而，使用者並不能真正地和目的端來作點對點的資料傳送，而是透過防火牆的解析、過濾之後才轉送至目的端。

2. 各項工作重點：

接下來解說每一項環節的主要關鍵之處。

a. Client 端的要求：

```
String s = "ask";
Socket skt = new Socket(ServerURL，port);
OutputStream out = skt.getOutputStream();
out.write(s.getBytes());
```

這裡主要是對某一端主機的通訊埠送出要求的訊息，我們利用串流的方式針對防火牆主機上提供的專用通訊埠來要求建立加密通道。

b. 監聽通訊埠：

```
ServerSocket ss = new ServerSocket(port);
skt = ss.accept();
```

利用建立 Socket 的方式來監聽某個通訊埠，等待遠端的要求，接收到遠端傳送的 Socket 之後，便可利用此 Socket 來得知遠端主機的 IP-Address、Hostname 等資訊。

c. 傳送 Agent：

```
AgletProxy proxy =
getAgletContext().createAglet(null，
"project.SlaveAglet "， getProxy());
destURL = new URL(String s);
destProxy = proxy.dispatch(destURL);
```

當接收到要求之後，防火牆便會創造一個 Agent 並且送出至要求端的主機上執行，送達遠端之後，透過 proxy*來控制 Agent 的活動(如：Agent 的 retraction、disposal 等)

*proxy：負責管控 Agent 的所有行為

d. 封包的處理：

有兩種封包，一是向外發送、一是往內接收，我們是利用網路串流來作封包的傳送及接收。(在這裡指的封包是一種稱為 socket 的高階抽象觀念)

<<接收>>

```
skt = ss.accept();
FileInputStream in = skt.getInputStream();
```

這段程式是當防火牆接受要求並傳送Agent至遠端主機之後，遠端透過message傳送來要求防火牆等待接收socket。防火牆再將接收到的socket內容串接到輸入串流中。

<<發送>>

```
skt = new Socket(destAddress，port);
decodeSend();(or encodeSend();)
```

在這裡會先建立目的端主機的socket,再加密或解密送出封包。

e. Agent的交談：

Agent之間可透過message機制來互相交談，但是兩者之間必須知道對方的ID。通常是由一方派送(dispatch)另一個Agent出去，這時候雙方便互相擁有彼此的ID，並可藉此達成交談目的。

```
if(message.sameKind("getReady"))
remoteProxy.sendFutureMessage(new
Message("accept"));
```

此交談目的在於通知對方下一步要作些什麼。

f. 加/解密：

在這個部分，任何的加/解密技術都可套用，像是DES、RSA、RC2 等等，由於專利的問題，我們是直接採用安全性普通的 DES 演算法技術。在封包送出之前會先串連到所謂的密文串流(Cipher Stream)加密，而接收端在收到封包之後，再將之串連至密文串流解密。而用來加/解密的 key 是防火牆端所給定的，它是在創造出 Agent 時所產生出來的 AgletID，由於 AgletID 是具有唯一性以及長達 16 個字元的不規則組合，所以適合拿來當作 key 使用。但是，對稱性金鑰的密文技術在安全上遠比非對稱性金鑰密文來得低，因為這是可以用暴力法來猜測 key 的內容。不過，因為 Agent 的安全性高，就算有惡意的 Agent 企圖偷取 key，也會因為自身的 AgletID 不為防火牆所送出的 Agent 的 AgletID 而失敗。最後，假若再加上第三者的認證，來證明雙方的身份正確與否再進行 Agent 的傳送，那麼安全性就相當嚴密，不過我們沒有將認證技術加進來，這是未來可以改進的地方。

g. 安全性檢查

在 Client 端送出要求、防火牆回應送出 Agent 之後，並不會馬上建立起兩端的加密通道，而是此 Agent 將安檢的程式送到 Client 端，對 Client 端掃描環境是否安全，其檢查包含 Client 是否開啓自己的防火牆，以及企業內部統一化的防毒軟體版本及病毒碼是否和 Server 端的版本一致，之後才會真正開始與防火牆互動。

3.6 實例說明

舉一個實例來說明我們的專題，這樣更能了解我們專題的實作部分，假設 Client 端欲傳送檔案到 Server 端(圖 3.3) 。

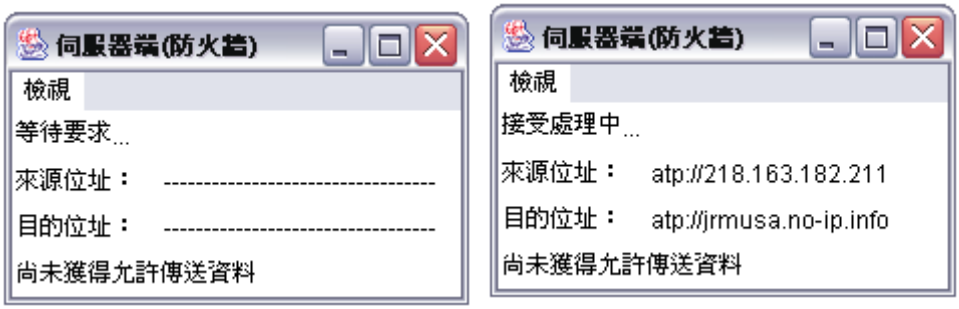

圖 3.3 防火牆端等待及接收 Client 請求

在 Client 端的介面中向防火牆端發出建立安全通道要求，當防火牆端收到請求時，便發出一個 Agent 給 Client 端來檢查 Client 的安全性(圖 3.4)

圖 3.4 安全性檢查視窗

安檢的過程中，若 Client 端的環境不安全(Client 端未開防火牆、防毒軟體或病毒碼版本太舊)，則防火牆會中止其連線，要求 Client 做更新的動作(圖 3.5)，若通過防火牆的安全性檢查，則防火

牆端會在 Client 端開啓傳輸介面(圖 3.6)，以讓 Client 端準備傳輸資料。

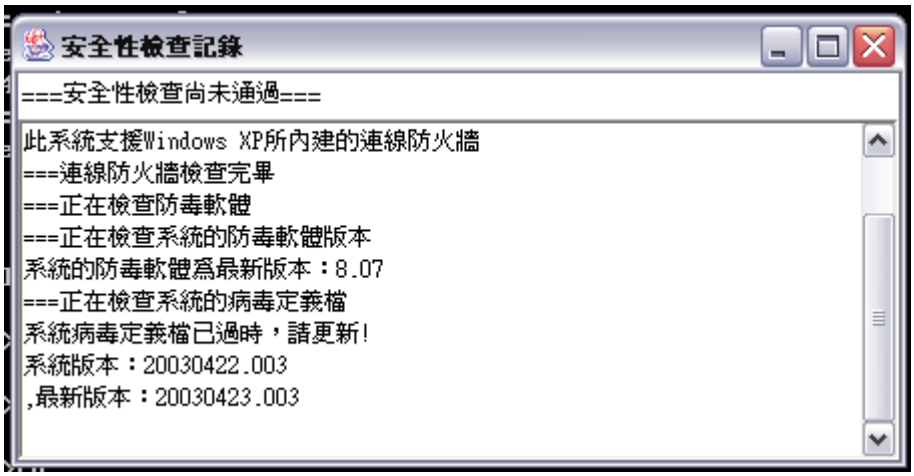

圖 3.5　Client 端環境不安全

圖 3.6　Client 端傳輸介面

而防火牆端也會準備將從 Client 端的檔案資訊傳送到 Server(圖 3.7)

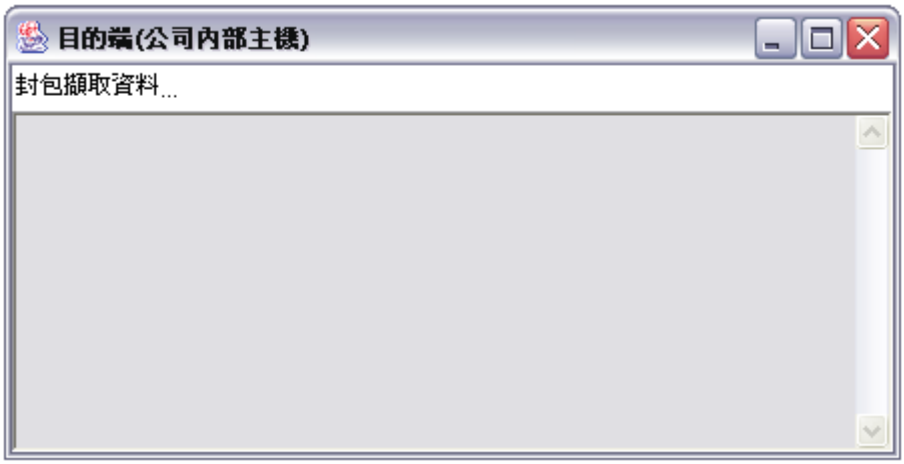

圖 3.7　防火牆端準備傳送至 Host 端

然後 Host 端便會接收到從 Client 端的檔案，而防火牆端也會記錄下整個傳輸檔案的過程(圖 3.8)。

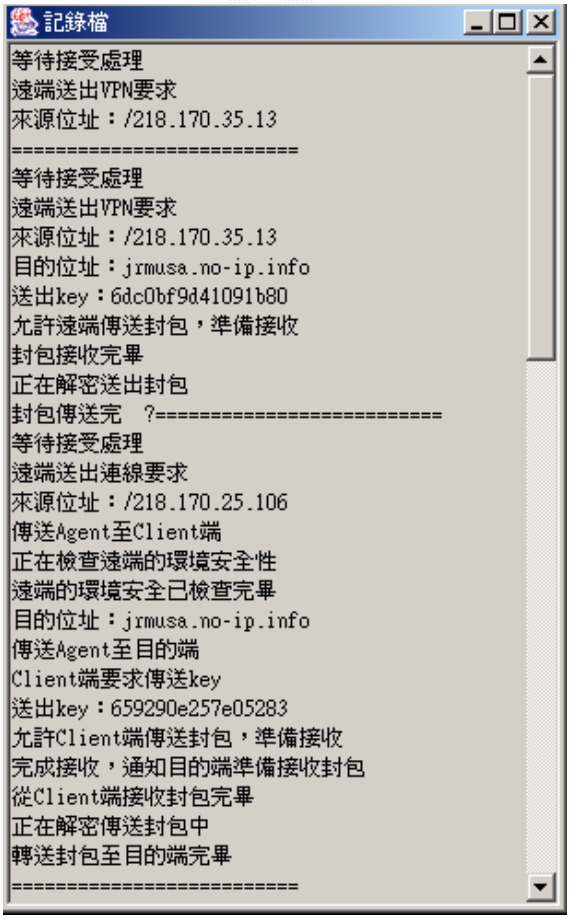

圖 3.8　記錄檔

然而在遠端傳送資料時，我們有分為二種模式，一種是非點對點的傳輸模式，此模式是將封包在防火牆端解密之後，再傳送到 Host 端(圖 3.9-1、3.9-2)：

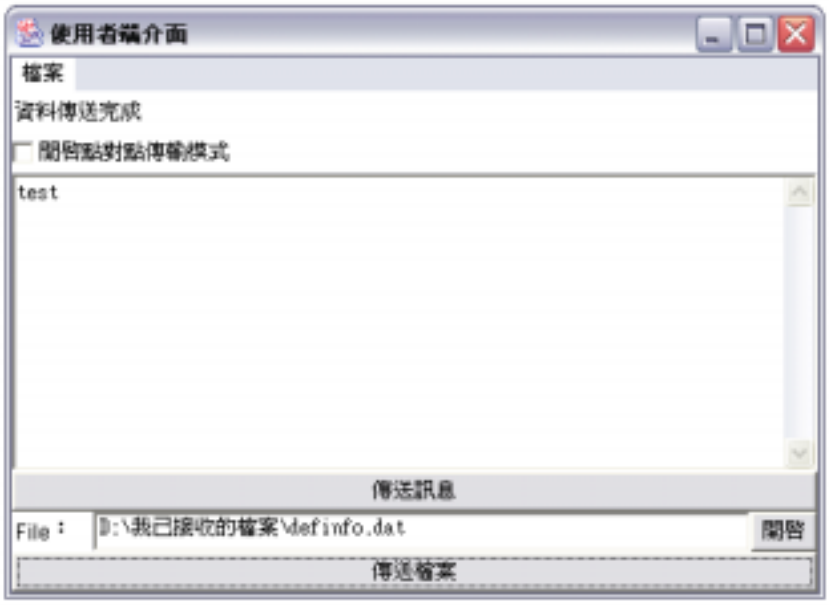

圖 3.9-1　遠端傳送資料(非點對點模式)

圖 3.9-2　目的端接收資料(非點對點模式)

而另一種傳輸模式是點對點的傳輸模式，其作法是防火牆端直接將 Client 端傳送過來的資料直接送到 Host 端，讓 Host 端自行做解

密的動作(圖 3.10-1、3.10-2)，在此只有傳送到 Host 端，並未做解密以分別出二者之間的差別，圖 3.10-1 中，第一個”test”字串為非點對點的傳輸，而在”test”字串後的亂碼，即為點對點的傳輸。

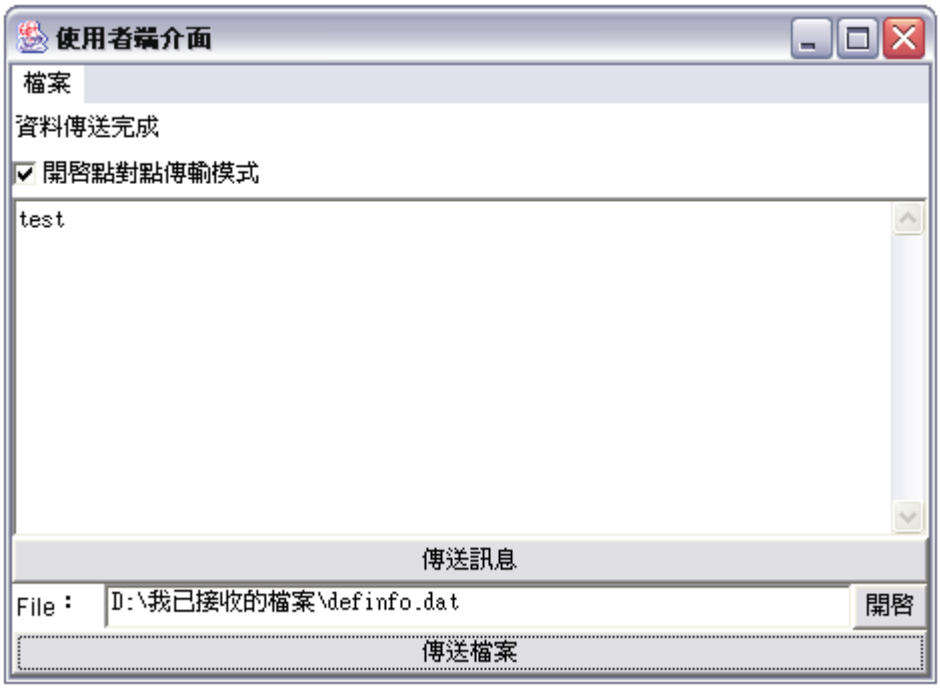

圖 3.10-1　遠端傳送資料(點對點模式)

圖 3.10-2　目的端接收資料(點對點模式)

第四章　總結

4.1. 結論與建議

遠端分散式安全主要著重在於傳輸通道以及 Client 端的環境安檢，傳輸通道是 Client 和 Server 之間的溝通管道，如果這個連線被不懷好意的駭客所利用，容易造成其內部的主機資料外流或被竊取及破壞，而使得公司的財產受到嚴重損失。因為讓 Client 端能夠存取 Server 端的資料，若 Client 端沒有相當的安全性保護，相對的也就是讓 Server 端暴露在危險的環境中，因此這二點是我們著重的要點。

我們利用 Mobile Agent 來實作安檢加密通道，因為 Agent 有遠端分散式處理及非同步處理的特性，Server 端使用 Agent 將安全檢查程式傳送到 Client 端的 run time system 裡執行，加上 Agent 的自主性，一方面可以將安檢機制從 Server 端配送到 Client 端，另一方面也能夠防止 Client 端的使用者企圖將安全檢查程式修改，達到其安全的目的，一些不法的使用者就難以竊取及修改傳送中的 Agent 資料。也因為在 Client 端多加了防火牆、防毒軟體及病毒碼的檢查，確保了 Client 端環境的安全性，進一步使得企業內部的主機得以降低被破壞的機會，因此我們算是成功的實作出遠端安檢的加密通道。

分散式的遠端加密通道在未來可以和防火牆及 Server 技術做更加緊密的結合，使得 Agent 的行動性在分散式的系統能夠做到更大的

發揮，在安全性方面也可以再加強對 Client 端安全環境的檢查機制，相信一定更能使加密傳輸有更高的利用性。

4.2. 專題所遇到的困難：

在我們實作專題時，曾遭遇到了下列的困難點:

1. 未曾接觸過的領域：

Mobile Agent 對我們來說是很陌生的，國內相關資料少之又少，必須到國外的網站或論壇找尋，剛開始只能從最簡單的 sample program 著手，後來漸漸地了解了整個 Aglets 的架構，也可以逐步測試專題的各項目標。

2. 環境設定複雜：

Aglets 並沒有完整的安裝程式，而是必須自行設定，另外像是 Server 端也需詳細設定 Security、domain 及 Agent 的執行權限，因此，整體環境偏向封閉的系統而非開放式。

3. Server 端的防火牆：

原本初步的構想是與防火牆作結合，也就是說在防火牆內常駐一 Agent 來檢查特定要求的封包，不過這又是另一項新的挑戰，對我們來說也是一個新的領域，因此，我們並沒有做到很完整的封包檢查。

4. Client 端的安檢機制：

我們的安檢程式可以檢查防毒軟體的版本、病毒碼、以及防火牆的啓用與否，但是並未真正與軟體作溝通，而是檢查、比對記錄檔(log)，以及記憶體中的常駐程式，因此這方面可以進一步的改善。

4.3.個人心得

吳國銓

在大學的生涯裡，我寫過大大小小不同的程式，用來交作業的、專為興趣寫的、或者參加比賽用的，這些都比不上專題程式的繁雜。從題目的訂定、找尋相關資料、研讀文件、撰寫程式、修改到完成，每個步驟都需要經過仔細的思考及討論，這讓我學到了跟交作業完完全全不同的經驗。

一開始的題目制定讓我傷透腦筋，不曉得從何著手，有一次題目幾乎底定了，但是由於受限硬體的關係，讓我們的計畫中止，害得原本投入的心力都浪費了，只好重頭開始。後來跟專題指導老師討論了很久，才決定研究 Mobile Agent，原本我沒信心可以做出來，因為題目偏向理論，再加上相關資料並不充足，一度想要放棄，不過到後來，在找相關資料研讀及請教學長之後，才慢慢有了初步的概念，於是想放棄的念頭也就慢慢消失了。

由於國內相關資料幾乎沒有，只能藉由網路找尋資料、以及閱讀 Mobile Agent 的原文書，剛開始很難下手寫一些簡單的程式，只能夠看看語法，連怎麼運用 Agent 都不是很清楚，到最後才慢慢地從小程式的原始碼學習，再思考如何將研究目的實作出來，這過程的確為我帶來很多的經驗。

另外，小組討論很重要，關係著整個專題的成敗，在討論的過程中，有時候會突發奇想，然後就有了很大的進展，或者在討論中，得到自己獨處時得不到的靈感。

大學裡的專題研究讓我學到了不同於念書的研究精神，以及團隊合作的重要性，雖然說自己並不是很滿意自己的研究成果，但是我得到了很多的經驗，這是最重要的。

陳重宏

剛開始要找專題時，我們都很有點盲目，不知道要做什麼樣的題目，後來找上了我們導師李維斌老師，他給了我們幾個方向，於是我們就開始找相關的資料，最初我們要做的題目是想在手機上實作出條碼，那時候我們就專研在 J2ME 和條碼的技術上，雖然做出了一個初版的條碼，但是放到手機中，拿去便利商店測試時，卻無法成功的掃瞄到我們的條碼，於是這個題目宣告失敗。

於是老師又給了我們另一個方向---Mobile Agent，也就是我們現在發表的這個題目，在做這個專題時，也是碰到了很多問題，因為Mobile Agent 算是個滿新的技術，在資料上我們能找到的東西並不像以前一樣那麼多，不過聽說有學長在做有關這方向的東西，於是向學長請教了很多有關 Mobile Aglets 的問題，學長也給了我們很多的意見，也借了一本 Aglets 的書，這對我們的專題有相當大的助益，一開使最大的難題算是在 Aglets 的使用上吧，我們花了滿多時間在了解與使用 Aglets 上，不過有了學長的幫助一切都還滿順利的，然而在程式的撰寫上也是遇上了很多 java 方面的問題,像是 I/O 串流、加密等等，對我們算是有點不熟悉的技術，不過二個人相輔相成收穫真是不少，不過當我們解決一個問題，就會又發生另一個問題，像是在網路的設定上，因為沒有固定的 IP 會抓不到來源，當我們改用定址的方式後，雖然可以通了，卻又出現安全性的問題，不過最後還是被我們一個個的解決了。

在這次的專題中，讓我學到了不少東西，像是團結合作的重要性，及如何解決分爭等等問題，專題是一個 team 而不是個人，所有的事都要以團隊為重，雖然其中不免會有很多的爭吵，但是最重要的是能一一的解決，完成最終的目標，這才是最重要的，所以專題對我的意義不只是技術上的增長，更是未來出社會前的一種磨練。

4.3. 專題進度

月份	2	3	4	5	6	7	8	9	10	11	12	1
尋找老師	●	●										
收集資料		●	●									
確立目標			●									
系統分析				●								
分配工作				●								
實作				●	●	●	●	●	●			
測試修改									●	●		
撰寫報告										●		

參考資訊

[1]D.B.Lange and M.Oshima，”Programming and Deploying Java Mobile Agents with Aglets”，Addison-Wesley，1998

[2]E.R. Harold，”JAVA I/O 技術 ”pp.273-329， O’REILLY，2001

[3]E.R.Harold，”Java 網路程式設計”， O’REILLY，2000