引用 SOAP 技術之行動代理人訊息系統
A SOAP Based Message-System for Mobile Agent

陳志達
成功大學電機工程學系
台南市大學路一號
andypony@turtle.ee.ncku.edu.tw

蔡尚榮
成功大學電機工程學系
台南市大學路一號
srtsai@mail.ncku.edu.tw

陳明輝
成功大學電機工程學系
台南市大學路一號
visper@turtle.ee.ncku.edu.tw

摘要

本文以研究如何運用 SOAP(Simple Object Access Protocol)建立一個提供不同 Mobile Agent 平台交換訊息的 Message Gateway，不同平台的 Agent 可以透過 Message Gateway 整合服務。由於 SOAP 結合 HTTP 與 XML 做為 PRC 形式的通訊，利用 SOAP 做為訊息交換的機制，可以得到不受異質系統限制與被廣為使用之 HTTP 的好處。我們也設計了提供 Java 的 Client 端 API(Application Programming Interface)，以存取 Message Gateway 上的訊息。同時我們引入兩個 Mobile Agent 平台，在這兩個平台上提出一個完整的應用，讓不同平台的 Mobile Agent 透過 Message Gateway 進行通訊，以驗證系統的嚴整與可用性。

Abstract

Mobile agent has got its attention both in academy and industry. There are more than fifty mobile agent systems all over the world and many related systems are still under construction. Without a mechanism for communication, all these heterogeneous agent systems may not be integrated to provide more powerful services.

In this thesis, the research objective is to apply SOAP(Simple Object Access Protocol) to build a message exchanging system called Message Gateway for different mobile agent platforms. Agents on different platforms can communicate one another via the Message Gateway to integrate service. SOAP supports RPC style communication based on HTTP and XML. Using SOAP as the basic mechanism of message has two important advantages. One is the ability to support heterogeneity, the other is the popularity of HTTP transportation support. We developed a set of Java APIs(Application Programming Interface) for clients to access our Message Gateway. Besides, two mobile agent platforms are applied to manifest the ability and usability of our Message Gateway.

Keyword : Mobile Agent, SOAP, Message Gateway, XML, RPC

1. Introduction

隨著網際網路的發達與 Web 技術的成熟發展，網路上的資訊已經由以往的資訊貧乏，轉變成為今日的資訊爆炸。想在茫茫網海中找到所需的資料，往往必須花費相當多的時間及人力。雖然，網路上不乏眾多的搜尋引擎，但每次的搜尋的結果總充斥著許多不相關的資訊，以目前的技術來說必須以人工的方式進行過濾。這樣的結果使得我們想在網路上找到所需的資訊必須付出更多的時間與精力。於是許多系統便採用 Mobile Agent[1][2]來解決這樣的問題，期望透過 Agent 來取代人類重覆性的行為，並藉由 Agent 遷移到資訊所在的站台處理資訊，以減低對網路頻寬的依賴與達到負載平衡的目的。但是，因為有許多不同的 Mobile Agent 系統存在，不同系統間要建立一個溝通管道似乎不是那麼容易。另一個嚴重的問題就是，目前的網頁設計以展現資訊為主，設計的考量主要以供人類閱讀為目的，並沒有考慮對 Agent 的可讀性，使得 Agent 的未來存在著許多的問號。

因應這樣的趨勢，我們規劃一個引用 XML(eXtensible Markup Language) [3]與 SOAP

（Simple Object Access Protocol）[4][5]整合的訊息系統，並且將 Web 上的資源以 XML 來描述，希望藉由這個系統能夠達成以下的目的：

1. 整合各家的 Mobile Agent 的系統服務，使得不同 Mobile Agent 平台間能有一個交流的管道。
2. 透過 HTML 結合 XML 來描述 Web 上的網路資源，使得 Web 由 Human readable 的型態，轉變為 Agent Friendly 與 Agent Readable。

當然，這樣的系統並非一定要用 SOAP 與 XML，可是藉由 SOAP 跨平台[6][7]的特性可以使 Agent 整合更多的資訊來源，也利用 SOAP 穿透防火牆的能力結合更多企業內的資源；另一方面，利用具格式標準、語意明確的 XML(Extensible Markup Language)技術來達成資源整合是初步的構想。

2. System Architecture

在我們系統的設計上，可分為兩個部分一個是 Message-Gateway，另一個是 Mobile Agent 的設計（參考圖 2-1）。

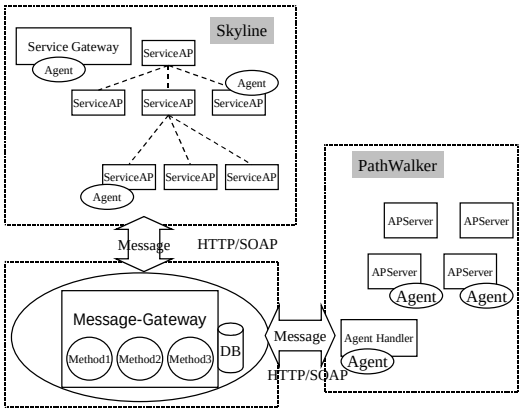

圖 2-1 系統整體架構圖

在不同Mobile Agent平台本身幾乎都具有不同的通訊機制，Agent 和 Agent 間的通訊都可以依系統的通訊機制達成。當不同 Mobile Agent 平台之間的 Agent 要達成彼此通訊的需要時，可以利用 RPC 或直接以 Socket 達成[8]，但是這些方法很難達成交換格式的標準化。

Message Gateway 的設計主要以提供不同 Mobile Agent 系統間的通訊，Mobile Agent 系統間則是利用 Message-Gateway 達到資源整合的目的，並引用 SOAP 作為通訊協定來簡化通訊的機制，以 XML 包裝訊息的內容作為訊息的標準格式。以下將分別介紹 Message Gateway 的設計與架構，Client 端 API 的設計，與兩個 Mobile Agent 系統架構的介紹。

2-1 Message Gateway 的設計

在我們探討 Message Gateway 的設計時，主要以 Mobile Agent 的使用為基本的考量，另外也將對系統可變尺度的設計作一個完整的規劃，以滿足系統未來擴充的需求[9]。

我們期望透過 SOAP 做為 Message 傳遞的方式，讓不同語言的 Agent 都能透過 Message Gateway 溝通，考慮到目前 Mobile Agent 系統多是以 Java 所開發，所以初期我們 Client 的 API 的設計也是以 Java 平台上為主。

本系統提供點對點的通訊，與群組的通訊功能。另外，考慮到 Mobile Agent 是遷移到各個不同的主機上執行，Mobile Agent 完成工作的時間很難達到一致，因此提供訊息的 Notification 機制也是必要的，以便讓訊息的交換機制更靈活。也正因為 Mobile Agent 可以游走在各個 Server 間，而且當未來 Agent 的數量相當可觀，必須由多個 Message Gateway 共同服務 Agent 的通訊時，當 Agnet 遷移到不同 Message Gateway 的範圍時，訊息也必須能準確的送達，因此追蹤 Agent 的位置也是相當必要的。在某些考量下(例如安全性的考量，必須分辨 Agnet 的權限時)，賦予 Agent 一個 ID 做為身份的認證也是必須的。

基於上述的考量，我們希望對 Mobile Agent 做訊息交換提供以下幾個需求：

1. 管理 Mobile Agent 的 ID，取得一個 Unique 的 ID 與歸還。
2. 追蹤 Agent 的位置與狀態。
3. 訊息的傳遞，可針對一個或多個 Agent 送出訊息。
4. 訊息的傳遞，可針對整個 Group 傳遞訊息。
5. 可以由 Message-Gateway 將訊息主動送給 Agent。
6. 在未來 Agent 數量增加時系統具有擴充性。

2-2 可變尺度之設計

我們設計所有的 Message-Gateway 均會擁有一個 Cluster Table，記錄相鄰近的 Message Gateway 位置、代號與從屬關係，Message Deliver(參閱 3-2 Message Deliver) 透過 Tracking Process 所建立的 Location Table（參閱 3-3 Process Tracking)，將訊息轉送給其他負責的 Message Gateway。[10]

Cluster Table 的內容包含了三個欄位，MG Serial Number、Location 與 Parent：

1. MG Serial Number：主要就是的 Agent ID 部分，前面十碼是 Message Gateway 的代號，後六碼是 Agent 的序號，在系統中 Message Gateway 即是代號加上序號為 000000(如：0000000123--000000)。
2. Location：用來記錄該 Message Gateway 的 IP 位置，當 MG Serial Number 是自己的序號時則用來 Parent 的 IP 位址。
3. Parent：記錄各個 Message Gateway 的父節點代號如果是最上層的 root 則此欄位是空白。

所有的 Message Gateway 的 Cluster Table 中必須包含兩項資訊，包括 Parent 的代號與位置，另外就是 child 的 Cluster Table，如此一來 Message Gateway 的 Cluster Table 才能明確的表示包含自己的 subtree 結構。

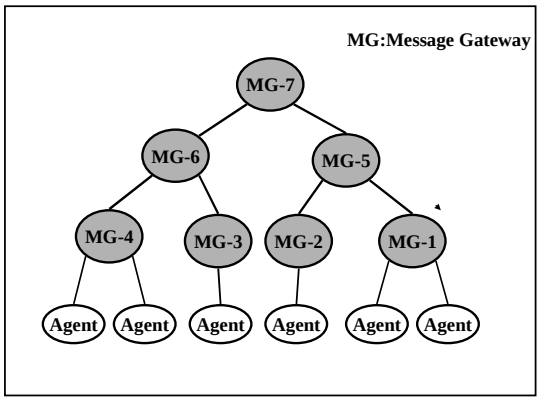

圖 2－2 Message Gateway Cluster Tree

3. Message Gateway 的架構

針對系統設計的考量，在 Message-gateway 的設計上分為四個主要的元件（參考圖 3－1 Architecture of Message-Gateway)：

1. ID Management：用來管理線上 Agent 的 ID，賦予與 Agent 一個 unique 的 Serial Number，與 ID 的使用時限管理，並記錄一些 Agent 的基本資訊。
2. Group Management：用來管理群組的設定，使用者可以利用加入群組獲取某一個 Channel 的訊息，或將多個 Agent 設定為訊息廣播的對象。
3. Message Deliver：負責訊息的發送工作，將訊息直接發送給 Agent，也可以將訊息轉送至其他的 Message Gateway；訊息的發送除了將訊息存入 Agent 的 queue，再由 Agent 主動取回訊息外，也可以在訊息到達時通知 Agent。
4. Process Tracking：追蹤 Agent 所在的位置與狀態，登錄 Agent 的所在位置，記錄 Agent 是否能接受訊息通知，與記錄 Agent 所在的 Message Gateway 代號。

透過這四個元件的運作，滿足系統的所需的通訊需求，將 Agent 間所要傳遞的訊息準確的送達。

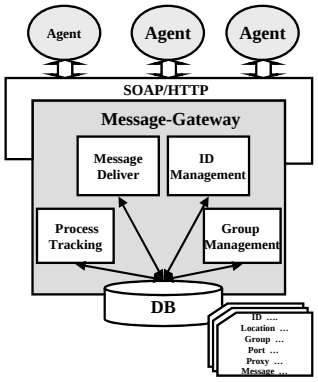

圖 3－1 Architecture of Message-Gateway

3-1 ID & Group Management

在系統中所有的 Agent 都必須取得一個 ID，做為與其他 Agent 溝通的代號，這些代號的管理與維護將會是個頭痛的問題。因此，我們必須規劃出一套方法來將這些 ID 回收，以利這些 ID 的重複使用。

在我們的系統設計中，應用 Message Gateway 來傳遞訊息的程式可以分成兩類：

1. Mobile Agent：在系統中會在各個站台間移動，完成交付的任務後就會消失(參考圖 3－2)。
2. Service Agent：長期在主機上提供服務的 Agent(參考圖 3－2)。

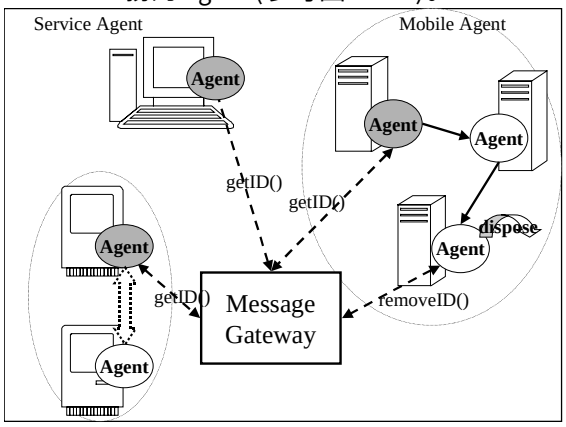

圖 3－2 Agent 取用 ID 示意圖

因此，當 Agent 向系統取得 ID 時即加以分類，所有 Agent 的預設值設定為 Mobile Agent，專為提供服務的 Agent 則必須向系統註冊為 Service Agent，系統管理者亦可視情況保留某些固定的 ID 給長期提供服務的 Service Agent，當 ID 用盡時的系統檢查各個 Agent 最後一次要求服務時間，當 Agent 超過時間未向系統要求服務而且其形式為 Mobile Agent 時，判定該 Agent 失效，應將 ID 釋出，若為 Service Agent 則予以保留。當系統管理者發現系統的 ID 已逐漸不敷使用時，此時 Message Gateway 也會因為使用者過多，也會造成系統效能變差，因此建議管理者可以參考 2-2 **可變尺度之設計**中所提之架構，將系統依照需要擴充成階層式架構或環狀架構，以分散系統的負荷。

群組管理方面，我們希望透過群組的設定，讓 Agent 與 Agent 之間建立一個傳遞訊息的廣播的 Channel，Agent 可以不必知道彼此的 ID 代號，可以對多個 Agent 發送訊息。Agent 可指定所有加入群組的成員，以簡化對多個 Agent 發送訊息。也可以選擇加入有興趣的群組，接收該群組的所有相關訊息。

3-2 Message Deliver

訊息的管理方面主要可以分為兩個部分來討論：

1. 第一個部分為 Message Gateway 內部各個 Agent 間訊息的交換，包含各個 Agent 的 Message Queue 的管理，與 Agent 新訊息的通知機制。
2. 另一個部分為 Agent 與不同 Message Gateway 的 Agent 間的通訊，包含 Agent 由一個 Message Gateway 管轄的範圍遷移到其他 Message Gateway 的管轄範圍，訊息的轉送機制，與各個不同 Message Gateway 上各個 Native 的 Agent 間的通訊機制。

在系統中所有的 Agent 都擁有一個 Message Queue 用來暫存多筆的訊息內容，當 Agent 向 Message Gateway 取回訊息後，所有被取閱過的訊息即會自 Queue 中清除。在 Message Queue 中每一筆訊息必須包含了訊息的來源、訊息的接收者與訊息的實際內容，當 Agent 向 Message Gateway 取回訊息時可以包裝成 XML 的訊息格式，再將訊息回傳給 Agent。

在系統中送訊息的機制，發送端可以選擇發送的方式，當發送端選擇一般的發送則訊息只會被加到指定的 Agent 的 Message Queue 中，即完成發送訊息的動作，待訊息的收方向 Message Gateway 詢問訊息時才將訊息自 Queue 中取出。若發送端選擇訊息必須 Notify 所有成員，那麼 Message Gateway 除了會將訊息存入各個 Agent 的 Message Queue 外，尚會依據 Agent 所登錄的 Port 以 TCP/IP 的 Socket 連線，向該 Agent 登錄的 Port 送出"You got a message!!"的訊息，Agent 如果收到訊息可以自行向系統取回訊息。

當然，系統也支援一次將訊息發送給多個 Agent，這包含了群組的通訊與指明多個 Agent ID 的通訊，透過 Group Management 建立的群組資

訊，查出所有群組 ID 的 List，再將訊息逐一交由 Message Deliver 中的 Delivery Process 將訊息存入 Message Queue。

訊息的接收機制，接收端也可以選擇接收的方式，最基本的方式就是由接收端主動向 Message Gateway 詢問，有訊息在 Agent 的 Message Queue 中系統即將訊息包裝成標準的格式傳回給 Agent。另外，Agent 也可以向系統登錄一個 Port，當發送端選擇該筆訊息必須 Notify 時，可以透過 TCP/IP 的連線通知接收端。考慮到 Agent 的位置可能處在防火牆後，防火牆的管理員不可能任意的讓使用者開啟 Port 監聽外部的封包，這個時候前面所說的 Notify 機制即無法使用，解決的方法我們採用透過改變 Message Gateway 的 HTTP 的 Connection Timeout 時間長度，將 Timeout 的時間延長。

接下來我們討論 Message Gateway Cluster 上的 Agent 間通訊的方式，設計上我們以 Message Gateway 服務群為樹狀結構與與環狀結構做為設計的考量（如下圖 3－3 Message Gateway Cluster(樹狀結構)與圖 3－4 Message Gateway Cluster(環狀結構)）：

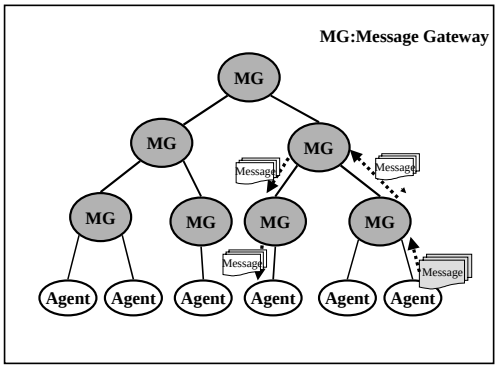

圖 3－3 Message Gateway Cluster(樹狀結構)

對於樹狀結構的 Message Gateway 服務最底層的子節點(Leaf)只需在 Cluster Table 記錄其父節點(Parent)的位置，當有位於其他 Message Gateway 的 Agent 的訊息時一律將訊息轉送至其父節點，由父節點依其 Cluser Table 將訊息轉送給其他子節點。在父節點的 Cluster Table 的內容則包含了其下的所有子樹 (subtree) 的 Cluster Table 與本身的父節點資訊，換句話說就是每個 Node 都完整的擁有一份包含本身的 subtree 結構，當一個節點收到外部的 Message Gateway 的訊息時可以明白的決定應該將訊息往哪一個節點送，只要不屬於其 subtree 的訊息一律交由父節點處理，若屬於其 subtree 的訊息則將訊息轉往負責的子節點。

這種方式對 Message Gateway 原生的 Agent 之間通訊並不會造成任何額外的負擔，而對由一個 Message Gateway 的管轄範圍遷移到另一個 Message Gateway 的 Agent 而言也只需要在遷移完成後透過 Tracking Process 將相關的訊息更新，發送訊息的 Agent 只需要將訊息往所屬的 Message Gateway 送出，Message Gateway 的 Message Deliver 機制會負責將訊息轉送到 Agent 所在的位置，不會有任何額外的負擔。

對於 Message Gateway 間的通訊機制，考慮到不同 Message Gateway 設計的簡化，與 Message Gateway 間也有可能有防火牆，我們讓 Message Gateway 間的通訊也透過 SOAP 交換訊息。如果在 Location Table 中存在該 Agent 的 ID 並沒有登錄 Proxy 的代號表示該 Agent 目前正在該 Message Gateway 的管轄範圍下運作，此時可以依照呼叫的 function 去 Notify 該 Agent 或將訊息存入 Message Queue。

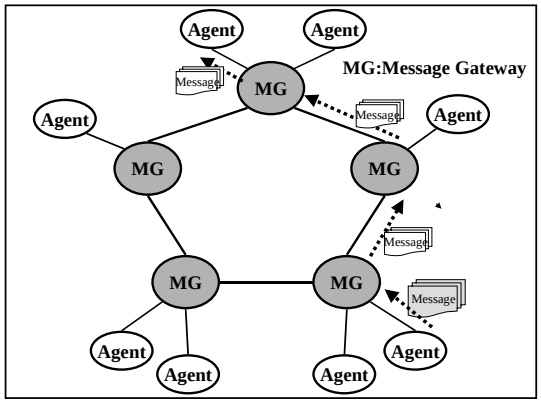

圖 3－4 Message Gateway Cluster(環狀結構)

對環狀結構的 Message Gateway 群（圖 3－4 Message Gateway Cluster(環狀結構)），也則是將樹狀的的結構設計變化而來，所有的相鄰 Node 既是互為 Parent 也是 Child，在 Cluster Table 中只記錄右側的 Message Gateway 的代號與位置資訊作為 Parent，依此類推形成一個環狀的結構。訊息的傳遞則是如圖所示逆時鐘方向轉送至 Agent 所在的 Message Gateway，再由 Message Gateway 將訊息放入 Message Queue 中或 Notify

該 Agent 有新訊息。

3-3 Process Tracking

Process Tracking 負責的工作主要有：

1. 追蹤 Agent 所在的位置與狀態，登錄 Agent 的所在位置。
2. 記錄 Agent 是否能接受訊息通知。
3. 記錄 Agent 所在的 Message Gateway 代號，當 Agent 遷移到其他 Message Gateway 的管轄範圍時，更新相關的 Message Gateway 的 Location Table。

系統中提供 agent 呼叫 setLocation()，以更新 Location Table 中 Agent 所在的位置，基本上 Message Gateway 只能被動的接收 Agent 提出更新位置的需求。未來，可以透過提供 Agent 更高階的 API 來簡化 Agent 遷移的代價。

系統中提供的 setPort()也是由 Agent 主動提出設定的需求，當 Agent 呼叫時更新 Location Table 中的 port，在此同時 Agent 必須監聽該 port 的訊息，藉此得知是否有新訊息，未來，提供可以更高階的 API 簡化 Agent 設計。

在此之前我們先說明 Proxy 的定義，在系統中所有的 Agent 都可以指定一個 Porxy, 而 Proxy 所對應到的是一個 Message Gateway。當 Agent 在其原生的 Message Gateway 上時 Proxy 就是這個指這個原生的 Message Gateway 的 ID，這個時候 Location Table 中的 Proxy 欄位則是空的，所有的訊息在 Local 的 Message Gateway 上即可處理;可是當 Agent 遷移到其他的 Message Gateway 的管轄範圍時, Proxy 指的就是這個新的 Message Gateway，這個時候其原生的 Message Gateway 就必須將這個 Message Gateway 的 ID 加入 Location Table 的 Proxy 欄位，當有該 Agent 的訊息時便轉送 Porxy，由 Proxy 來處理該 Agent 的訊息。

接下來探討 Agent 在各個 Message Gateway 群間遷移的追蹤，與路徑維護：

在 2-2 **可變尺度之設計**中曾經談到，利用 Cluster Table 的設定使得 Message Gateway 間訊息的傳遞存在樹狀的結構中，當 Agent 遷移到其他 Message Gateway 管轄的範圍後，立即呼叫 Local 的 Message Gateway 的 setProxy()，此時 Local 端的 Message Gateway 會將 Agent 的 ID 加入 Location Table，並且同時向其 Parent 送出 setProxy()的要求，當 setProxy()的要求送達涵蓋 Agent 來源的 subtree 的 root 時, setProxy()的要求將轉而向 Agent 原生的發送, 這些非 Local 端會將 Local 的 Message Gateway 的 ID 做為該 Agent 的 Proxy 值加入 Location Table，這個時候 Proxy 的設定即算完成(參考圖 3－5 Set Proxy after Agent Migrate(1))。

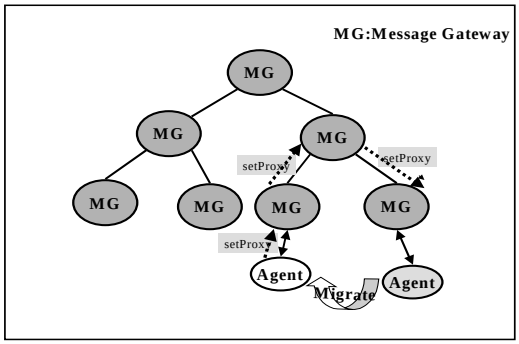

圖 3－5 Set Proxy after Agent Migrate(1)

接下來 Agent 再次遷移到另一個 Message Gateway 的管轄範圍，同樣的 Agent 呼叫 Local 的 Message Gateway 的 setProxy()，此時 Local 端的 Message Gateway 會將 Agent 的 ID 加入 Location Table，並且同時向其 Parent 送出 setProxy()的要求，當 setProxy()的要求送達涵蓋 Agent 來源的 subtree 的 root 時, setProxy()的要求將轉而向 Agent 原生的 Message Gateway 發送。在以上的過程中 Message Gateway 會比對 Agent 是否已經設定過 Proxy，如果曾被設定 Proxy 表示這個 Message Gateway 是涵蓋之前的 Proxy 和 Agent 來源的 subtree 的 root，那麼便會對舊有 Proxy 的路徑送出 removeProxy()將相關該 Agent 的資料移除，並且繼續對 Agent 原生的 Message Gateway 的路徑發送 setProxy()的命令(參考圖 3－6 Set Proxy after Agent Migrate(2))。

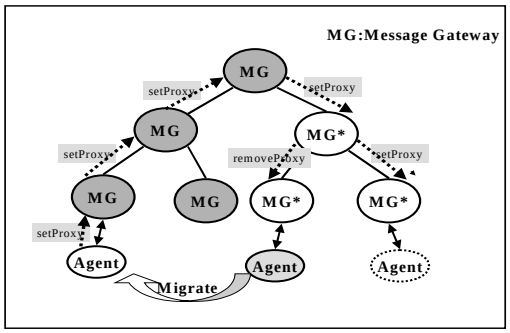

圖 3－6 Set Proxy after Agent Migrate(2)

當 Agent 完成任務結束工作後，即向 Local 的 Message Gateway 送出 removeProxy()的訊息，此訊息將沿著先前建立的路徑反向逐一送出 removeProxy()，將所有關於該 Agent 的資料移除。(參考圖 3－7 Remove Proxy after Agent Migrate(3))

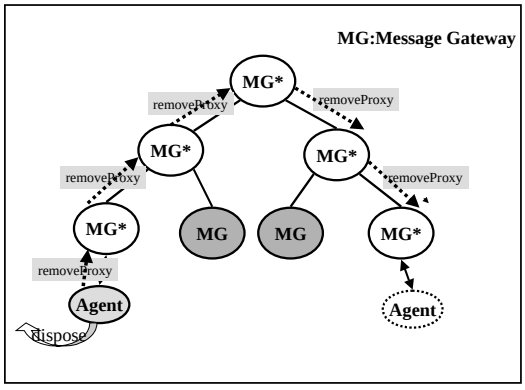

圖 3－7 Remove Proxy after Agent Migrate(3)

此一機制當然也可以應用在前一小節所述的環狀的 Message Gateway 群，操作方式與樹狀結構是一樣的。

3-4 Client 端 API 之設計

我們已提供 Message Gateway 的 Client 端 API，目前提供 Client 端使用的 API 為利用 Apache-SOAP[11]在 Java 語言上所開發，主要考量 Java 語言具多方面的優勢, 以及 Mobile Agent 的設計平台以 Java 為主, 未來將視環境變化開發不同語言的 API，使用者亦可利用微軟所開發之 SOAP Toolkit[12]與 XML Parser 建立與 Message Gateway 溝通的管道。

5 應用實例

我們初步以旅館資訊做為一個實驗性的對象，假設許多旅館分別加盟在不同 Mobile Agent Service Provider 下，這些 Mobile Agent 系統之間彼此無法得到一個標準的訊息交換管道，因此我們引入 SOAP Based 的 Message Gateway 做為不同平台間溝通的管道，透過 Message Gateway 整合各個系統中的資訊，並將最有用的訊息呈現給使用者。

使用者可以透過 Skyline 上的 Service Gateway，將使用者需求交給 Agent，Agent 則再將需求交給 Message Gateway 中 ServiceAgents 的群組中的成員，Agent 則負責收集 Skyline 的旅館資訊。最後整合由 ServiceAgents 群組成員所傳回的旅館資訊，再將結果 Email 給使用者。

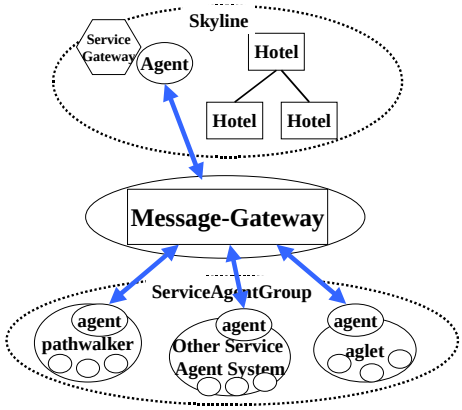

圖 5－1 Global View of Hotel Locator

所有的 Mobile Agent 系統只要加入 ServiceAgents 的群組，這些 Mobile Agent 就可以將自己的 Service 讓其他 Mobile Agent 系統分享, 在這個部分我們在 Pathwalker 已經實作出一個 Mobile Agent 的 Hotel Locator 來提供服務。

而想要取得其他 Mobile Agent 系統資源的 Agent，只要將需求以 XML 表示，並將訊息送給 ServiceAgents 的群組成員，隨後能提供服務的成員會將結果傳送到該 Agent 的 Message Queue 中。

這樣的設計，Service 的提供者可以自由的加入服務的群組，Service 的要求者也可以輕易的取得相關的訊息。

透過這樣的應用，我們希望能實際驗證系統中 Agent ID 的管理與取得，點對點通訊與群組通訊，與系統 Notification 的功能，藉由這些基本的能力進一步證明，透過我們設計的 Message Gateway 做為不同 Mobile Agent 平台間交換訊息的管道是可行的方案。雖然系統具備了上述的一些優點特性，系統目前已知的一些問題，這些問題包括：

結論

我們利用 SOAP 來建立 Message Gateway 做為 Mobile Agent 的基礎通訊架構，包括點對點的通訊與群組通訊的需求，與訊息主動通知等功能，以滿足 Agent 的基本通訊需求。

系統中所提供的基本的功能包括了幾項，我們也驗證了這些功能的運作：

- 點對點通訊：提供 Agent 透過 Message Gateway 將訊息交給指定的 Agent，達到點對點的通訊功能。
- 群組通訊：可以讓 Agent 將訊息以廣播的方式交給群組的成員，此部分功能與 Channel-based 的通訊類似。
- Notification：Message Gateway 透過 Socket 通知 Agent 有新訊息，再由 Agent 主動取回訊息，此項功能可能受限於防火牆而沒有完整的實現。

在我們的系統中透過 SOAP 做為訊息交換的格式，滿足基本的 Agent 的通訊需求外，還包含了以下的特點：

- 跨平台的使用：Message Gateway 滿足在不同的 Mobile Agent 平台上的可用性，即使不同語言的 Mobile Agent 也可以透過 Message Gateway 達到通訊的需求。
- 訊息內容標準化：系統中所有的訊息均以 XML 來描述，可以建立一個標準的訊息交換格式。
- 良好的系統擴充性：考慮到系統的未來的擴充能力與通訊效率，我們也提出一套可行的架構，滿足未來的擴充需要。

我們的系統確實的滿足以上幾項的特點，訊息的傳遞也能穿透防火牆，對於系統的安全性分面，SOAP 雖然可以透過 SSL 傳遞，但是訊息內容的加密與認證並沒有確切的規範，在我們的系統中，對於訊息的安全性部分，希望保留給各個系統依照不同的應用需要自行規劃。

參考文獻

[1] David Chess, Colin Harrison, and Aaron Kershenbaum, ” Mobile Agents：Are They a Good Idea?”, IBM Research Report, March 1995

[2] Robert Gary,David Kotz,Saurab Nog,Daniela Rus and George Cybenko, ”Mobile Agent: The next Generation in Distribute Computing”,0-8186-7870-4/96 1997 IEEE

[3] “Extensible Markup Language(XML)”, Tim Bray , Jean Paoli C. M. Sperberg-McQueen,,“Extensible Markup Language (XML)1.0 “, http://www.w3.org/TR/REC-xml, February 1998.

[4] “Simple Object Access Protocol(SAOP)1.1”,W3C Note 08 May 2000,http://wwww.w3.org/TR/SAOP

[5] Aaron Skonnard ,”SOAP：The Simple Object Access Protocol” ,msdn online Microsoft Internet Developer

[6] Don Box ,”A Young Person’s Guide to The Simple Object Access Protocol：SOAP Increases Interoperability Across

Platforms and Languages” ,msdn online Microsoft Internet Developer

[7] Simon Simeonov , “Dividing into the SOAP specification” , http://www.sys-con.com/xml/archives/0105/simeonov/index.html

[8] Amy L. Murphy and Gian Pietro Picco, “Realiable Communication for Highly Mobile Agent” , 0-7695-0304-3/99 1999

[9] “Communication Concepts for Mobile Agent System” ,J. Baumann et al. ,WorkShop MA'97 LCNS1219

[10] Sashi Lazar, Ishan Weerakoon and Deepinder Sidhu,” A Scalable location Tracking and Message Delivery Scheme for Mobile Agents”, 0-8186-8751-7/98 1998

[11] “Apache-SOAP Version 2.0：User’s Guide ”,August 2000, http://www.alphaworks.ibm.com/tech/soap4j

[12] Rob Caron, “Develop a Web Service：Up and Running with the SOAP Toolkit for Visual Studio”, http://msdn.Microsoft.com/msdnmag/issues/0800/webservice/webservice.asp