支援移動性用戶之穩定溝通機制

陳忠信　　施宏達　　黃冠霖
朝陽科技大學網路與通訊研究所
41349 台中縣霧峰鄉吉峰東路 168 號
Tel: (04) 23323000 ext.7243
{jschen26, s9530618, 9430620}@cyut.edu.tw

摘要

網際網路上網路位址(IP)具有區域性，導致傳統的網路程式設計以指定固定的 IP 為導向。但目前蓬勃發展的可攜性裝置，會因人漫遊在不同的無線網路，而導致裝置的 IP 不停的改變。這讓傳統網路程式無法正常運作。有鑒於此，為了讓程序間的資料傳遞能支援移動性，我們提出一個新的機制，讓程式設計師可以在不考慮移動性的條件下來發展軟體，而發展出的軟體卻具備移動性。這個機制中包含了三個功能，能保障傳遞出的資料不會因使用者的移動性而遺失。依照此機制，我們完成了一個新的 Java 類別 (class)，來驗證我們提出的機制是可行的。根據效能評估可知，使用者通訊時改變區域網路，也能成功的接收訊息，因此本系統為支援移動性用戶的穩定溝通機制。

關鍵詞: Agent、User Mobility、Terminal、Mobility、JADE、Stability。

一、介紹

網際網路與無線網路快速發展，隨著移動通信的高速成長，可攜式裝置大眾化，人們利用可攜式裝置相互通訊，造成新一波的通訊革命。傳統的網路程式設計以指定固定的 IP 為導向，使用者移動時，區域網路的變更，即改變 IP，造成無法持續與其它使用者通訊及訊息遺失等問題。

可攜式裝置的移動會讓軟體設計者產生很大的困擾，就傳統的軟體設計而言，採用的是 End to End 的方式，即本地程序必須先指定遠端程序的 IP 位址才能進行通訊，而不會考慮到裝置的 IP 位址會隨時改變的問題。而根據 Mobile IP 上定義移動性的型態可分為：

(a) 終端機移動性─程序執行所在的終端機設備移動。
(b) 使用者移動性─使用者移動到原先所在的地點位置不同。
(c) 服務移動性─程序間通訊的資料內容能穿透各種不同的異質網路。

以上這三種移動性的型態都會造

成程序間無法持續通訊，只要在移動時本地端程序無法即時掌握遠端程序的 IP 就無法持續的通訊，造成資料接收上的遺失。

例如: 一台位址屬於 IP 網段 A 的 PDA，從網路 A 移動到具 IP 網段 B 的網路。因為接收端改變所屬的網路位址，所以來源端無法成功完成傳送資料。就過去的研究中，雖然利用 Mobile IP 協定的概念可以解決 IP 改變所造成的問題，但是由於 Mobile IP 協定必須硬體上有所支援，每個區域網路都必須具有一個路由器(Router)，去做為轉送資料的代理人，達到資料內容能準確地轉交到可攜式裝置上。但並不是每個網路都具備此代理人設備，如果要使用去解決 IP 改變所造成的問題，所有的區域網路都必須安裝支援。對於現實中所需花費的金錢成本需求太高。

然而，程序與其他程序間互相通訊的過程中，不管何時何地，任何程序離開所在的可攜式裝置移動到另一個可攜式裝置，都可以與其他程序繼續通訊，並擁有在先前可攜式裝置上有所的資料，服務能不中斷地持續進行下去，遺憾的是，傳統網路程式設計語言沒有支援程序間移動性的函式庫，因此比較難去撰寫可支援使用者移動性的程式。

本論文提出一個資料內容傳送保存方法來解決程序間 IP 改變所遇到的問題，為了讓程序間的通訊能夠透過軟體設計就能有效的支援移動性，並讓程式設計者能簡易的設計使用，提供一個更高階的介面，自訂一個新類別，此類別裡面有三個方法，讓本地程序在使用上可以不受本地端或遠端移動性的影響，準確有效率地和遠端程序通訊。有效解決 IP 改變或主機改變時，還能保有內容接收上的正確性，讓通訊的資料內容不會因使用者移動造成遺失，達成無縫隙的空間。

這篇論文其他的內容安排如下，在第二章節會說明整個溝通機制的架構，以及改變網路的流程。第三章節說明在主機中資料的保存方法。第四章節介紹所使用的方法及程式片段範例。第五章節以效能估評的模擬數據來證明所提出的方法是支援移動性用戶的穩定溝通機制。

二、溝通機制

系統架構結合 JADE 代理人平台，主要分成伺服器和使用者，利用單一伺服器服務多個使用者。

為了讓使用者間能達到資料內容接收上不遺失和隨時獲得所有的資料內容，當使用者通訊改變區域網路時，也能成功的接收訊息。因此提出一套溝通機制，圖一為溝通機制的架構圖，使用者在同一個可攜性裝置上在各個網段之間移動，改變網路位址。首先需要開啟一臺不具移動性的伺服器，即可讓可攜式裝置之間及可攜式裝置與伺服器間利用代理人來傳送資料。而主要的架構分為四個階段:

(1) 註冊(登入):當使用者在任一可攜式裝置上啟動系統時，會先向伺服器註冊。
(2) 通訊:當程序 A 及程序啟動後，即可互相傳送資料。

(3) 確認回覆:當接收的程序,成功接收資料封包後，將回覆一個確認的訊息，給傳送的程序

(4) 備份:接收的程序成功接收封包後，將封包複製至伺服器當成備份資料;而傳送的程序收到確認回覆訊息後，也將自己的資料複製至伺服器備份。

圖二為溝通機制的流程:

(一)註冊(登入)的流程:

(1a) 使用者 A 進入系統先向伺服器註冊(登入)。

(1b) 使用者 B 進入系統先向伺服器註冊(登入)。

(二)通訊的流程:

(2) 在資料未接收完成的情況下，使用者 B 移動至下一個區網服務範圍，或在另一個可攜式裝置登入，使用者 B 在進入系統時，仍先向伺服器登入。

($2a_1$) 使用者A傳送一筆資料給使用者B。當 B 完成接收此筆資料時會回傳確認訊息給使用者 A。相反地，當使用者 A 在任何情狀下沒收到使用者 B 回傳的確認訊息時，判斷使用者 B 未完成接收資料，每隔 N 秒會傳送一次，直到使用者 B 收到為止。

($2a_2$~$2a_n$) 使用者 A 在任何情狀下沒收到使用者 B 回傳的確認訊息時，判斷使用者 B 未完成接收資料，每隔 N 秒會傳送一次，直到使用者 B 收到為止。

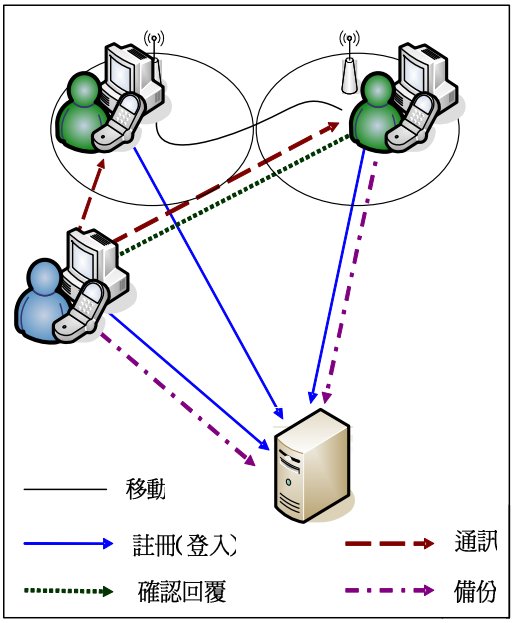
(三)確認回覆的流程:

(3) 使用者 B 完成接收使用者 A 的資料後，回傳確認訊息給使用者 A。

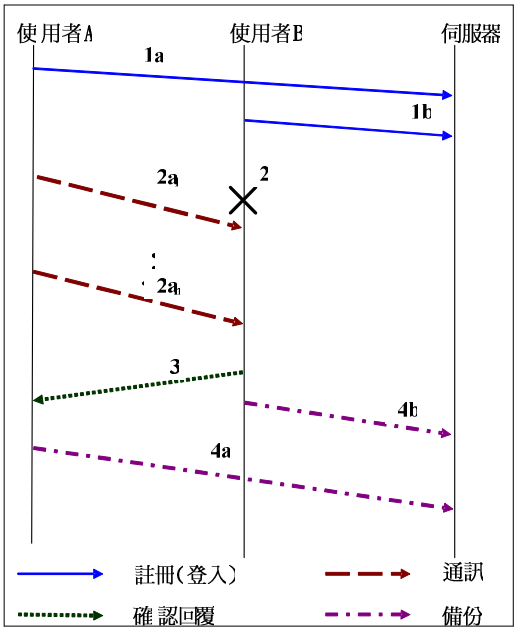
(四)備份的流程:

(4b) 使用者 B 完成接收資料後，複製到伺服器備份。

(4a) 使用者A收到使用者B回傳確認訊息後，將發送給使用者B的資料複製到伺服器備份。



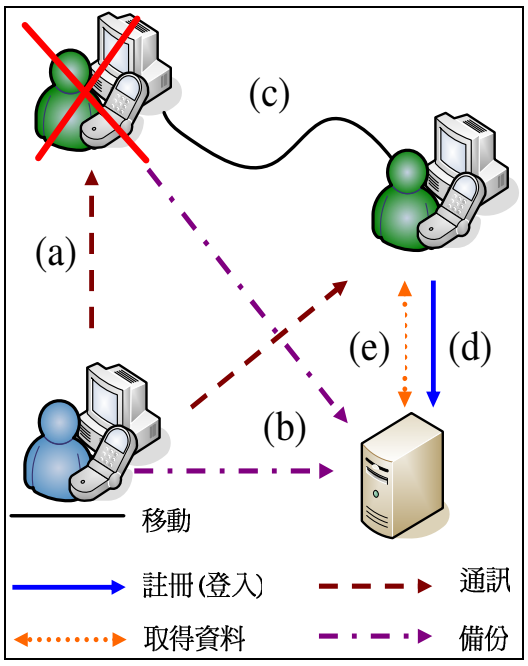
圖一 溝通機制架構圖



圖二 溝通機制流程圖

三、資料內容保存方法

當使用者 A 和使用者 B 進行通訊時，各自所在的主機會將該資料內容儲存下來，以便日後可用於查詢使用或讓程式設計者可以自行設計處理該資料。但當使用者 A 或使用者 B 離開儲存資料內容的主機，而在另外的主機上登入進行未完的通訊時，並不會把資料內容一併帶過去，而是留在舊的主機上。所以當使用者想在新的主機上查詢資料內容就無法獲得。



圖三 資料內容保存流程圖

本地程序和遠端程序之間必須能達到資料內容接收上不遺失，並能隨時獲得最新的資料內容，當使用者離開目前的主機，在另一個主機上登入時，需取得使用者在離開的主機上所有的資料。圖三為使用者在不同的可攜性裝置上登入的處理流程:

(a) 當使用者 A 和使用者 B 啟動後，即可互相傳送資料。

(b) 使用者 A 每傳送一筆資料，在確認使用者 B 收到後，就會去伺服器更新備份資料內容，相反地使用者 A 每收到一筆資料內容也會去伺服器做更新備份的動作。如果使用者 A 傳送一筆資料給予使用者 B，而沒有獲得使用者 B 的收到確認，則每隔 N 秒會傳送一次，直到使用者 B 收到為止。

(c) 當使用者離開目前所在的主機 B 繼而結束使用者 B，而在使用者移動到主機 B'上後，重新登入。

(d) 使用者 B 先去伺服器註冊後就可繼續和使用者 A 進行通訊。

(e) 使用者向伺服器請求屬於使用者 B 之前的資料內容，並將其帶回目前所在的主機 B'。

四、實作

我們設計一個新類別，讓程式設計者在不具備撰寫網路程式知識相關背景下，也能方便地的撰寫網路程式。整個系統架構結合 JADE 代理人平台，主要分成伺服器和使用者，利用單一伺服器服務多個使用者。啟動 JADE 代理人平台和伺服器後，使用者可在任何 Java 所撰寫的程式中去呼叫 Ncc 類別的 register( )、send( )和 receive( )方法，以下針對這三種方法做說明：

(a) Ncc.register(localId, serverIp)：

在使用 send()方法發送訊息之前，必須先讓程序向伺服器註冊，需傳入兩個字串，第一個字串代表本地的程序 ID，並用此 localId 向伺服端做一註冊；第二個字串代表伺服器的 IP，因為上述提到必須啟動伺服器，而伺服器的位置會因為使用者不同而有

所不同，所以先指定伺服器的位置。為了讓使用者間的程序能互相通訊，必須為每個程序指定一個 ID，用來作為區別彼此的依據。

(b) Ncc.send(localId, remoteId, content)：

用來傳送內容到本地程序和遠端程序間共用的 buffer 空間。此方法需傳入三個字串，第一個字串代表本地程序 ID；第二個字串代表所要通訊的遠端程序 ID；第三個字串代表所要通訊的內容。

(c) Ncc.receive(localId, remoteId)：

用來接收遠端程序傳送過來的內容，此方法需傳送兩個字串，第一個字串代表本地程序 ID，第二個字串代表遠端程序 ID。代表要去讀取哪個本地程和遠端程序共用的 buffer 內的內容。

圖四為說明傳送端和接收端程程序使用新類別的方法。在傳送內容程式範例中，一開始先新增一個 Ncc 的物件，並宣告四個字串，在這邊我們分別使用 localId、remoteId、content 和 serverIp 去代表，程式設計者可自行定義；然而必須先做 register()本地程序的 ID 和指定 Server 的位置，我們使用“User A”去指定本地程序的 ID, Server 的位置為”196.128.7.17”。當完成上述的步驟後，即可在下面程式的任何位置去呼叫 send()方法和遠端程序溝通。在這邊表示“User A”這個程序要和遠端“User B”程序通訊，內容為“HI HI!”。

在接收內容程式範例中，一開始先去新增一個Ncc的物件，並宣告兩個字串，在這邊我們分別使用localId和remoteId去代表，程式設計者可自行定義；之後可在下面程式的任何位置去呼叫receive()方法，即可得到需要的內容。在這邊表示要抓取“User A”和“User B”這兩個程序共用Buffer裡的內容。

```
public class sendclass
{
    public static void main(String argv[])
    {
        Ncc ncc = new Ncc();
        String localId ="User A";
        String remoteId="User B";
        String content="HI HI! ";
        String serverIp="163.17.21.94";
        ncc.register(localId,serverIp);
        /*
                program body
                                */
        ncc.send(localId, remoteId, content);
    }
}
```

傳送內容程式範例

```
public class receiveclass
{
    public static void main(String argv[])
    {
        Ncc ncc = new Ncc();
        String localId ="User B";
        String remoteId="User A";
        String serverIp="163.17.21.94";
        ncc.register(localId,serverIp);
        /*
                program body
                                */
        ncc.receive(localId,remoteId);
    }
}
```

接收內容程式範例

圖四 Ncc類別程式範例

利用這個新的類別，簡易於過去的網路程式，程式設計者不需要每次指定 ip 及 port，也不需使用 socket 來檢查是否要接收訊息，只需要簡易的

使用 register()方法來註冊、send()方法和其他程序通訊及 receive()方法來接收訊息，大大的改善程式設計者撰寫程式的難易度。

此類別利用JADE去實作一個代理人系統，藉由各程序所在的終端機設備上的代理人能隨時掌握本地和遠端程序的位置。當本地程序從一個網路區域移動到另一個網路區域時，常駐在本地程序的代理人就會去告知JADE平台最新的位置，在藉由JADE平台去通知其他遠端程序。

```
protected void check()
{
    hostAgent = InetAddress.getLocalHost();
    newIP=hostAgent.getHostAddress();
    if (!oldIP.equals(newIP))
    {
        Runtime rt = Runtime.getRuntime();
        name = getLocalName();
        checkpoint = 1;
        String cmd = "cmd/c java restart"+name;
        Process p = rt.exec(c);
        Thread.sleep(1000);
        DFService.deregister(this);
        System.out.println("terminating……0");
        doDelete();
    }
}
```

A.偵測 IP 位址改變程式

```
Thread sort2 = new Thread(new sendFile(id));
sort2.start();
sort2.join();
```

B.傳送備份資料檔案程式

```
Thread sort5 = new Thread(new rebuildFile());
sort5.start();
sort5.join();
```

C.儲存備份資料檔案程式

圖五 偵測IP改變、傳送、儲存資料程式

圖五列出部份的程式內容，圖五 A 是用來管理 IP 位址改變的 JADE 程式片段，當偵測到 IP 位址改變時，會去重新向 JADE 註冊，本程式是採用多執行緒方式撰寫，為了符合物件導向程式設計的特性。在圖五 B 是將傳送資料片段的程式，當程序 A 和程序 B 在進行通訊時，如果需要將資料檔案傳送到 Server 或其他程序做一備份的動作時，只要呼叫 sendFile()即可執行。此執行緒在傳送資料前會先將資料做一壓縮的動作，之後開啟一個 TCP 的 Socket 將資料傳送到遠端備份。而在接收資料檔案端，會先將其資料檔案完整接收下來。圖五 C 中呼叫 rebuildFile()執行緒，去將其解壓縮後儲存。圖五 C 當資料檔案接收完畢後，呼叫 rebuildFile()將其檔案解壓縮，並存在 historyArvhive 資料夾底下存放。

五、效能評估

於此章節中，對溝通機制中，傳送成功率、所需花費的時間、使用者移動性加以分析探討。

模擬的行為，由傳送端程序傳送資料到接收端。程序進入區網後，先至伺服端登入，當程序間傳遞資料，接收端未接收或未完全接收資料前即離開區網，則傳遞資料失敗，傳送端會在一段等待時間後重新傳遞，直到接收端成功接收資料為止。當成功傳送和接收資料後，則會到伺服端做一備份。

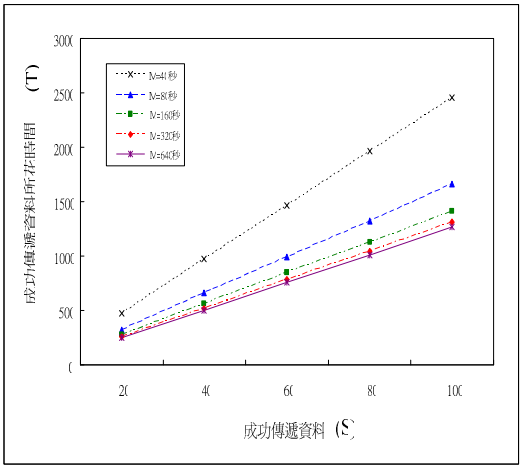
模擬環境的參數定義如下:

S: 總共需成功傳遞的資料數目。

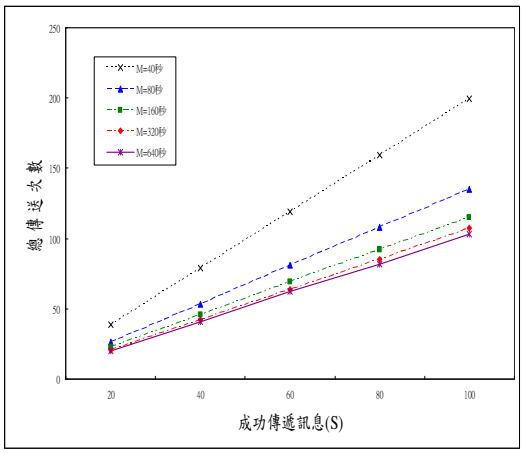
T: 成功傳遞 S 筆資料所花費時間。

M: 平均在一區網停留時間,在模擬的過程中,程序在一區網的停留時間由 Random 程序產生。

C: 平均傳遞一資料的週期,在模擬的過程中,程序間一程序的資料傳遞週期由 Random 程序產生。



A 成功傳遞資料所花時間 (T)



E 總傳送次數

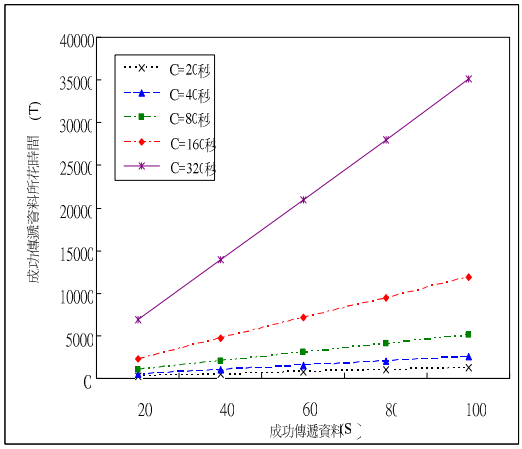
圖六 資料傳遞的平均週期固定 (C=20秒)

當程序間平均傳遞一資料的週期(C)固定時,分析成功傳遞資料(S),所花的時間(T) 及總代送次數(成功率)。如圖六 A 所示,當程序間平均傳遞一資料的週期(C)平均為 20 秒,程序由一個區網服務時間平均為 M=40 秒成功傳遞 20、40、60、80、100 次資料,所花費的時間分別為平均 475.84、974.27、1468.34、1962.72、2454.92 秒; M=80 秒成功傳遞 20、40、60、80、100 次資料,所花費的時間分別為平均 321.77、660.33、992.66、1322.98、1660.61 秒;M=160 秒成功傳遞 20、40、60、80、100 次資料,所花費的時間分別為平均 276.84、564.43、847.7、1127.64、1414.59 秒;M=320 秒成功傳遞 20、40、60、80、100 次資料,所花費的時間分別為平均 258.11、516.44、785.29、1042.91、1312.98 秒; M=640 秒成功傳遞 20、40、60、80、100 次資料,所花費的時間分別為平均 245.75、503.29、760.08、1006.36、1264.27 秒。

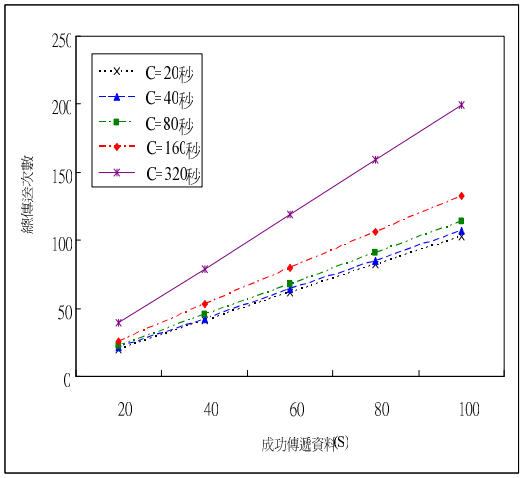
如圖六 B 所述,當程序間平均傳遞一資料的週期(C)平均為 20 秒,程序由一個區網服務時間平均為 M=40 秒成功傳遞 20、40、60、80、100 次資料,傳送端總共傳送的次數(成功率)分別為平均 38.67、79.08、119.24、159.42、199.55 次; M=80 秒成功傳遞 20、40、60、80、100 次資料,傳送端總共傳送的次數(成功率)分別為平均 26.15、53.62、80.72、107.84、135.09 次; M=160 秒成功傳遞 20、40、60、80、100 次資料,傳送端總共傳送的次數(成功率)分別為平均 22.52、45.96、68.99、92、115.02 次; M=320 秒成功傳遞 20、40、60、80、100 次資料,

傳送端總共傳送的次數(成功率)分別為平均 21、42.02、64、85.03、107 次;M=640 秒成功傳遞 20、40、60、80、100 次資料，傳送端總共傳送的次數(成功率)分別為平均 20、41、62、82、103 次。

因此，當平均溝通頻繁時(C)固定時，移動性(M)愈頻繁，使得接收端改變區網的可能性增加、成功傳遞資料的機率減少、傳送端重傳的次數提高，造成所需要花費的時間(T)線性的增加。



A 成功傳遞資料所花時間(T)



E 總傳送次數

圖七 區網平均的停留時間固定(M=640 秒)

當程序平均在一區網停留時間固定時，分析成功傳遞資料(S)，所花的時間(T)及總代送次數(成功率)。如圖七 A 所述，當程序平均在一區網停留時間(M)固定為 640 秒，不同的平均傳遞一資料的週期(C)，C=20 秒成功傳遞 20、40、60、80、100 次資料所需花費時間為平均 245.19、502.64、761.99、1007.17、1263.61 秒;C=40 秒成功傳遞 20、40、60、80、100 次資料所需花費時間為平均 490.42、981.56、1494.87、1987.59、2502.9 秒;C=80 秒成功傳遞 20、40、60、80、100 次資料所需花費時間為平均 998.35、2042.76、3085.66、4130.16、5174.06 秒;C=160 秒成功傳遞 20、40、60、80、100 次資料所需花費時間為平均 2317.44、4724.04、7126.48、9452.69、11860.33 秒;及 C=320 秒成功傳遞 20、40、60、80、100 次資料所需花費時間為平均 6872.45、13921.32、20970.77、28020.41、35087.4 秒。

如圖七 B 所述，當程序平均在一區網停留時間(M)固定為 640 秒，不同的平均傳遞一資料的週期(C)，C=20 秒成功傳遞 20、40、60、80、100 次資料所需傳送資料的次數分別為平均 20、41、62、82、103 次;C=40 秒成功傳遞 20、40、60、80、100 次資料所需傳送資料的次數分別為平均 21、42、64、85、106.98 次;C=80 秒成功傳遞 20、40、60、80、100 次資料所需傳送資料的次數分別為平均 22、45、68、91、114 次;C=160 秒成功傳遞 20、40、60、80、100 次資料所需傳送資料的次數分別為平均 26、53、79.95、106.04、133 次;及 C=320 秒成功傳遞 20、40、60、80、100 次資料所需傳送

資料的次數分別為平均 39、79、119、159、199.1 次。

因此，當程序平均在一區網停留時間(M)固定時，平均傳遞一資料的週期(C)時間愈頻繁(短)時，成功率愈低，傳送端重傳的次數也相對的提高。

六、結論

在點對點的溝通中，使用者移動時，會造成使用者因進入不同區域的網路而改變 IP，造成無法持續與其它使用者通訊及訊息遺失等問題。程序移動時所產生的主要問題就是 IP 改變。對此提出一個新的機制來達成穩定的通訊，不必改變大環境的硬體架構，即可解決 IP 改變所造成的通訊障礙。

我們實作了一個新類別來完成此機制，讓程式設計者在不具備撰寫網路程式知識相關背景下，也能方便地的撰寫網路程式。而完成的程式，使用者所在的本地程序可以不受移動性的影響，方便地和遠端程序通訊，達到資料內容的不遺失。

我們提出的機制，不僅讓程式設計師可以在不考慮移動性的條件下來發展軟體。根據效能評估的模擬數據可得知，在使用者通訊時進入不同區域的網路而改變IP，也能成功的接收訊息，因此本系統為支援移動性用戶的穩定溝通機制。

參考文獻

[1] A. Carrillo-Ramos, J. Gensel, M. Villanova-Oliver, and H. Martin, “PUMAS: a framework based on ubiquitous agents for accessing Web information systems through mobile devices,” 20th Annual ACM Symposium on Applied Computing - Handheld Computing, vol.2, pp. 1003-1008, March 2005.

[2] C. Perkins, “Mobile IP,” IEEE COMMUN. Mag., vol. 35, pp. 84-99, May 1997.

[3] E. Chen, D. Sabaz, and W. A. Gruzer, “Wireless distributed systems with JADE,” 2004 IEEE International Conference on Systems, Man and Cybernetics, vol.1, pp. 989-993, October 2004.

[4] E. Cortese, F. Quarta, and G. Vitaglione, “Scalability and Performance of JADE Message Transport System,” Proceedings of the AAMAS Workshop on AgentCities, 2002.

[5] G. Beuster, B. Thomas, and C. Wolff, “Ubiquitous Web information agents,” 14th European Conference on Artificial Intelligence, August 2000.

[6] G. Caire, “JADE tutorial-jade programming for beginners,” Version: JADE3.3, March 2005.

[7] G. Caire, LEAP user’s guild, Version: LEAP 3.3, March 2005.

[8] K. Mohammadi, and H. Hamidi, “An Approach to Fault-Tolerant

Mobile Agent Execution in Distributed Systems," The First IEEE and IFIP International Conference in Central Asia on Internet, pp. 26-29, September 2005.

[9] M. R. Lyu, X. Chen, and T. Y. Wong, "Design and Evaluation of a Fault-Tolerant Mobile-Agent System," IEEE Intelligent Systems, vol. 19, pp. 32-38, September-October 2004.

[10] P. Misikangas, and K. Raatikainen, "Agent migration between incompatible agent platforms," Proceedings of the 20th International Conference on Distributed Computing Systems, pp. 4-10, April 2000.

[11] T. C. Lech and L. W. M. Wienhofen, "AmbieAgents: a scalable infrastructure for mobile and context-aware information service," Proceedings of the 4th international joint conference on Autonomous agents and multiagent systems, pp. 25-29, July 2005.

[12] X. Meng, and H. Zhang, "An efficient fault-tolerant scheme for mobile agent execution," 1st International Symposium on Systems and Control in Aerospace and Astronautics, pp. 19-21, January 2006.

[13] Z. M. Zeng, B. Meng, and Y. Y. Zeng, "An intelligent agent-based system in internet commerce," Proceedings of the 4th International Conference on Machine Learning and Cybernetics, vol.1, pp. 299-303, August 2005.