國立交通大學

資訊工程學系

碩士論文

線上即時傳呼系統的設計與效能分析研究

The Study of Design and Performance Analysis of

Presence and Instant Messaging System

研 究 生：翁仕全

指導教授：吳毅成　教授

中 華 民 國 九 十 四 年 七 月

線上即時傳呼系統的設計與效能分析研究

The Study of Design and Performance Analysis of Presence and Instant Messaging System

研 究 生：翁仕全　　Student：Shih-Chiuan Weng

指導教授：吳毅成　　Advisor：I-Chen Wu

國 立 交 通 大 學
資 訊 工 程 學 系
碩 士 論 文

A Thesis
Submitted to Department of Computer Science and Information Engineering
College of Electrical Engineering and Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in

Computer Science and Information Engineering

July 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年七月

線上即時傳呼系統的設計與效能分析研究

研究生：翁仕全　　指導教授：吳毅成

國立交通大學 資訊工程學系

摘要

線上即時傳呼系統已經廣泛被大眾使用，由於使用者必需與伺服器保持連線，才可以即時傳送及接收當時的訊息，但是因為伺服器所能使用的連線有限，傳輸效率也需要保持在一定範圍之內，所以當使用者人數增加時，一般都是使用更大量的伺服器來提供服務以解決問題，但是成本相對提高很多，維護也較困難。

本論文希望研究現有線上即時傳呼系統的架構，設計高效率、可延展的線上即時傳呼系統架構，並對各種架構進行效能分析，找出各種架構正常運作下所能承受的最大人數。

The Study of Design and Performance Analysis of Presence and Instant Messaging System

Student: Shih-Chiuan Weng Advisor: I-Chen Wu

Department of Computer Science and Information Engineering

National Chiao Tung University

ABSTRACT

Presence and Instant Messaging (PIM) System is used popularly. Every user of a PIM system must keep connections with the server to send and receive messages. Since the maximum socket number of the server is limited, if the user number increases, they will increase the server number to solve this problem. But the cost will raise, and it is hard to manage.

We will study the architecture of PIM System, and design a highly efficient and scalable Presence and Instant Messaging System. Simulate and analyze the performance and scalability of the designed PIM system architectures to find the maximum user number for PIM system architectures.

誌謝

首先要感謝我的指導教授，吳毅成博士，由於他不厭其煩的細心指導，這篇論文才得以順利完成。

此外特別要感謝汪益賢學長和徐健智學長，在研究的過程中給予我許多寶貴的意見和指導。實驗室裡一起奮鬥的夥伴志祥、德彥以及學弟俊彬、育嘉、怡良、承翰的協助。在我的研究期間，給了我相當多的鼓勵，陪我度過這段最值得回憶的學生生活。

最後，我要感謝我的父母、弟弟、妹妹，在我的求學生涯中給了我最大的支持和照顧。謹以此論文，獻給我最摯愛的家人。

目錄

圖表目錄

第一章　緒論

1.1 研究目標

隨著網路的發展，任何人都能輕鬆的與其他地區的人取得聯繫，線上即時傳呼系統的出現，更是拉近了人與人之間的距離，使得人們能夠即時與線上的朋友交換訊息，比起以往的網路電子郵件及電子看板更為快速。此類軟體往往需要服務大量的使用者，又得即時的傳送訊息，因此軟體的效能便十分重要。

現今著名的線上即時傳呼系統大多由公司企業所研發，考量市場的競爭，較少做公開的線上即時傳呼系統效能研究分析，考慮使用者隱私的問題，也缺乏對線上即時傳呼系統使用者訊息的研究。

因此，本篇論文的目標在於線上即時傳呼系統效能研究分析，希望讓伺服器能支援愈多使用者愈好，以節省線上即時傳呼服務提供者的費用，也可大大提升服務的能力。

1.2 基本定義

線上即時傳呼系統(Presence and Instant Message System)是一個隨著網路發達而發展出來的系統，像 MSN[7][17][21]、ICQ[5][21]、Yahoo Messenger[16]，主要是提供即時訊息的交換，像是線上好友的狀態、聊天訊息，或是傳輸檔案、影像、語音交談等等。

1.3 論文大綱

在本論文第二章的背景說明裡，介紹線上即時訊息系統的基本運作方式。第三章介紹我們所設計的一些線上即時訊息系統的架構。第四章介紹我們如何以這些架構去進行模擬實驗，由實驗數據去分析，找出較適合線上即時訊息系統的架構。第五章是結論與未來的展望。

第二章　背景說明

本章介紹線上即時傳呼系統(Presence and Instant Message System)的背景以及使用情境。

2.1 線上即時傳呼系統的背景

近幾年來，網際網路蓬勃發展慢慢的改變了我們的生活，使得資訊的傳遞更為快速，無遠弗屆。一般來說，網際網路的應用可分為兩個種類：

一、 搜尋資料：

世界各地的電腦經由網際網路連接起來，我們可以很輕鬆的利用網際網路取得需要的資料，也可以將我們的資料分享出去。也因此發展出許多著名的搜尋引擎，讓人們可以更輕鬆的找到自己想要的資料。

二、 傳遞訊息：

網際網路提供了一個很方便的通訊管道，如網路電子郵件(E-mail)、電子看板(BBS)、網路聊天室等等軟體已成為大多數人與朋友聯繫的方法，取代傳統的書信甚至電話，因為利用網際網路溝通不僅快速、方便，而且便宜。

但是在現代快速的生活步調下，網路電子郵件無法即時獲得回應，電子看板、網路聊天室又需要雙方約定好到同樣的地點才可以進行訊息交換，所以新的通訊方式-線上即時傳呼，便逐漸開始被人們所使用。

線上即時傳呼系統中，使用者都可以擁有屬於自己的一份好友名單，當使用者 A 上線時，伺服器便會通知 A 的好友 A 上線的訊息，也會告知 A 有哪些好友正在線上，當雙方都在線上，便可以開啟一個視窗來進行傳送與接收訊息的動作。

2.2 線上即時傳呼系統的使用情境

而系統為了要達到讓使用者知道好友是否在線上以及傳送訊息給線上的好友，需要實作三種功能[3][4]：

一、 訂閱好友的狀態：

當使用者連上伺服器的時候，可以訂閱自己想知道狀態的對象，伺服器便會通知使用者這些人的狀態。如圖 2-1，A 向伺服器訂閱 B、C、D、E、F 的狀態。

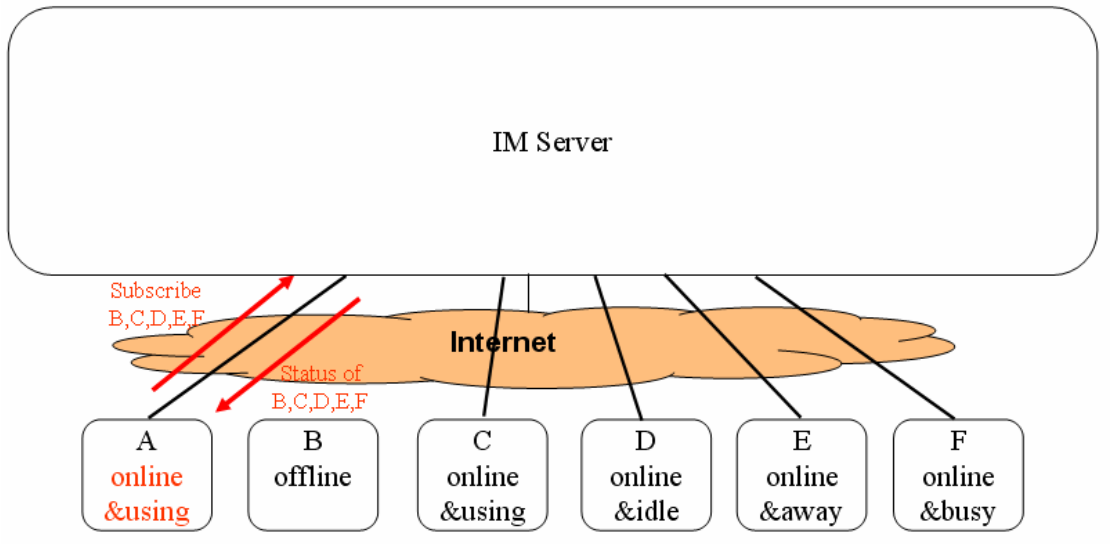

圖 2-1 訂閱好友的狀態

二、 通知訂閱者所需要的資訊：

當被訂閱者狀態改變的時候，伺服器會主動通知訂閱者。如圖 2-2，B、C、D、E、F 有訂閱 A 的狀態，當 A 狀態改變時，伺服器便會通知 B、C、D、E、F。

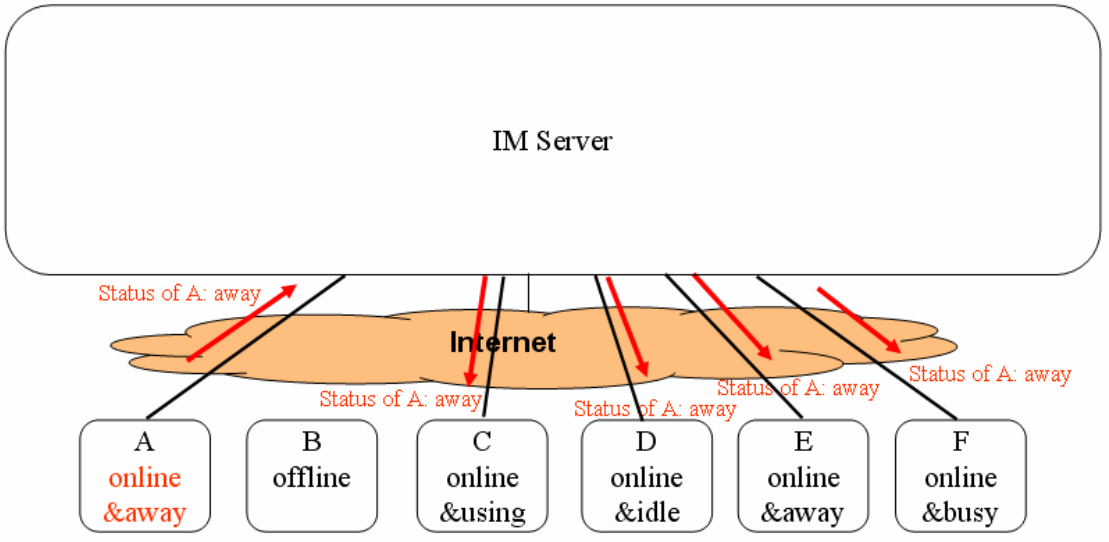

圖 2-2 通知訂閱者所需要的資訊

三、 傳送訊息：

使用者可以透過伺服器傳送即時訊息給其他線上使用者，其他人也同樣能傳送即時訊息給自己。

而對使用者來說，基本上會有四種行為：

一、 登入：

當 A 使用者進入系統時，系統會將 A 所訂閱的使用者資訊傳給 A，而將 A 的上線資訊傳送給有訂閱 A 資訊的使用者。如圖 2-3，A 有訂閱 B、C、D、E、F 的狀態，E、F 有訂閱 A 的狀態，當 A 進入系統的時候，系統會傳送給 A 他所訂閱的 B、C、D、E、F 使用者資訊，並且將 A 的上線資訊傳送給有訂閱 A 狀態的 E、F 使用者。

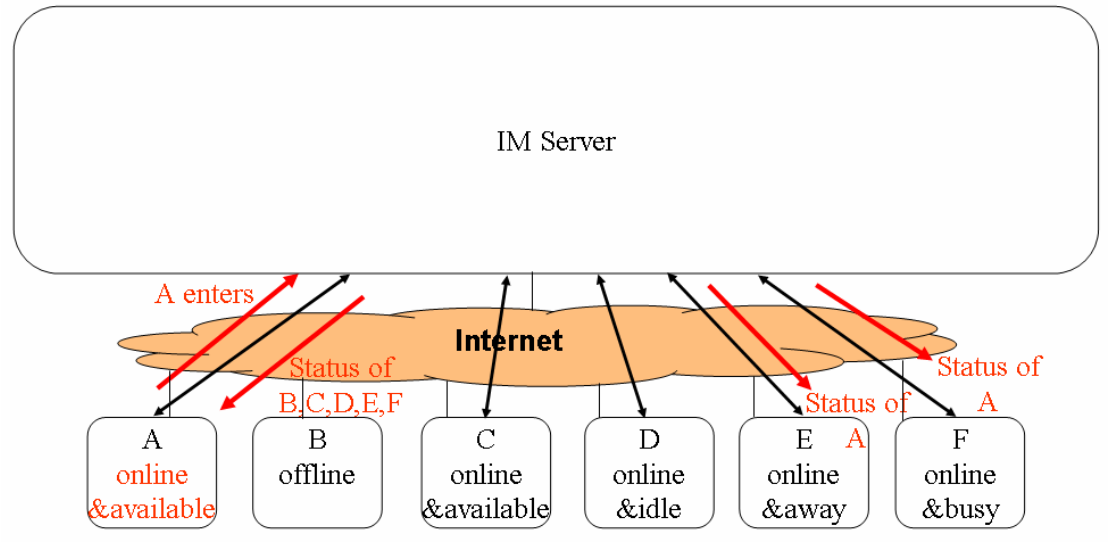

圖 2-3 使用者登入

二、 登出：

當 A 使用者離開系統的時候，系統會將 A 的離線資訊傳送給有訂閱 A 狀態的使用者。如圖 2-4，E、F 有訂閱 A 的狀態，當 A 離開系統的時候，系統便會通知 E、F。

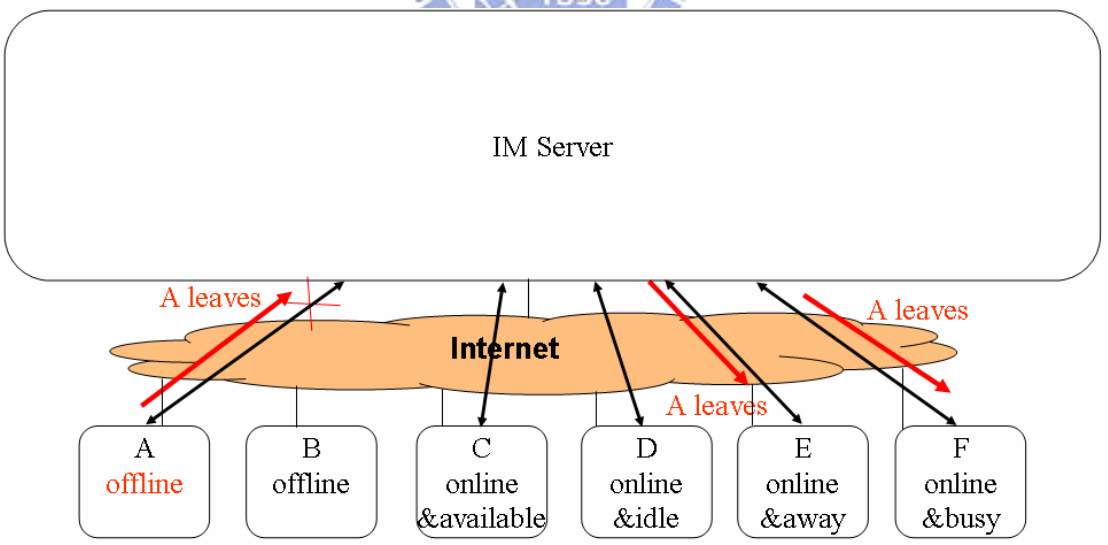

圖 2-4 使用者登出

三、 傳送訊息：

使用者透過伺服器傳送訊息。如圖 2-5，A 透過伺服器傳送訊息給 E。

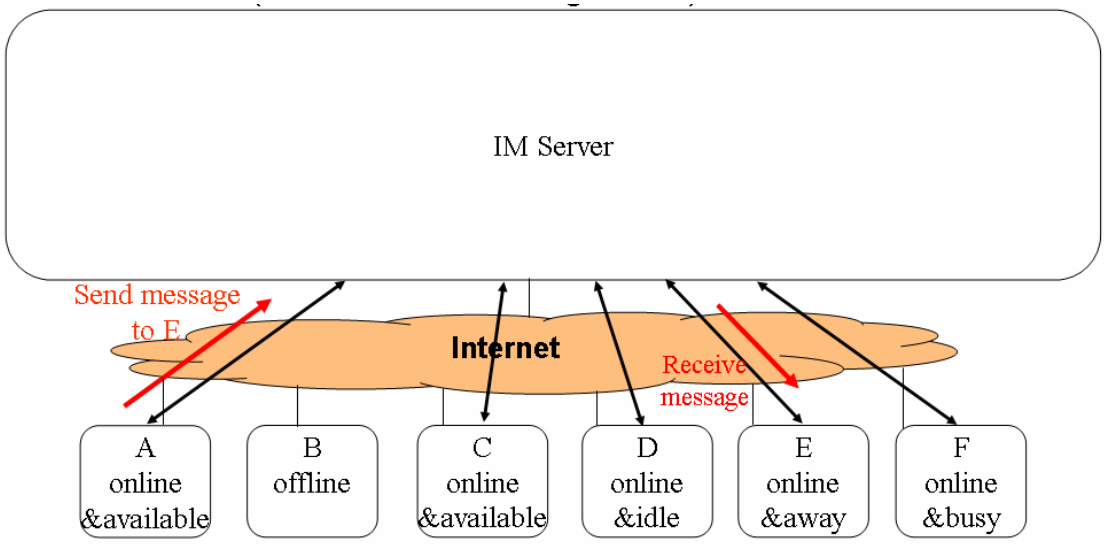

圖 2-5 傳送訊息

四、 改變狀態：

當使用者改變自己的狀態(線上、離線、忙碌、離開)，伺服器會通知訂閱者。如圖 2-6，E、F 有訂閱 A 的資料，當 A 將狀態改成離開，伺服器便會通知 E、F。

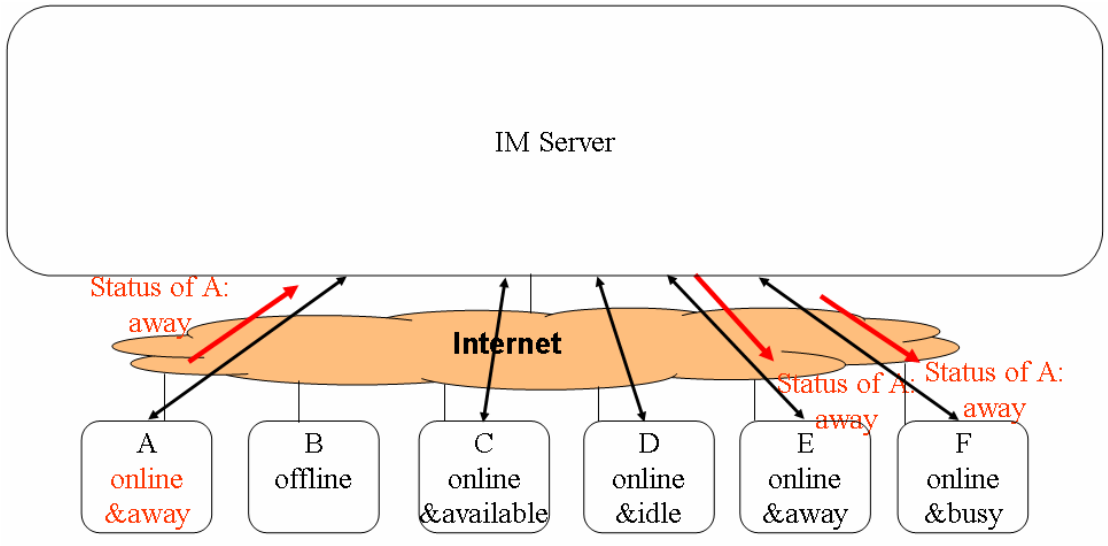

圖 2-6 改變狀態

2.3 線上即時傳呼系統的發展狀況

目前關於線上即時傳呼系統的研究[19]，多半都是關於使用者行為的分析。這些研究是使用訊息的記錄來分析，調查上班族等對象使用線上即時傳呼系統的行為模式 [10]。

當今較著名的線上即時傳呼系統有 MSN Messenger、ICQ、Yahoo Messenger、YamQQ、Skype 等軟體，表 2-1 是 2001 年美國的線上即時傳呼系統使用情形的調查結果[22]，可以看到較多人使用的是 AOL[1][20]，其次是 MSN Messenger、Yahoo Messenger、ICQ，這些系統功能已經相當完善，傳輸檔案，語音視訊交談等功能都可透過線上即時傳呼系統做到。但是由於考量到市場的競爭，較少做公開的線上即時傳呼系統的系統效能分析。

AOL	46%
MSN Messenger	20%
ICQ Chat	8%
Yahoo Messenger	13%
Other Protocols	13%

表 2-1 2001 年美國線上即時傳呼系統使用情形

2.3.1 ICQ

根據 ICQ 所公布的資料[18]，可推測出 ICQ 的系統構造如圖 2-7。

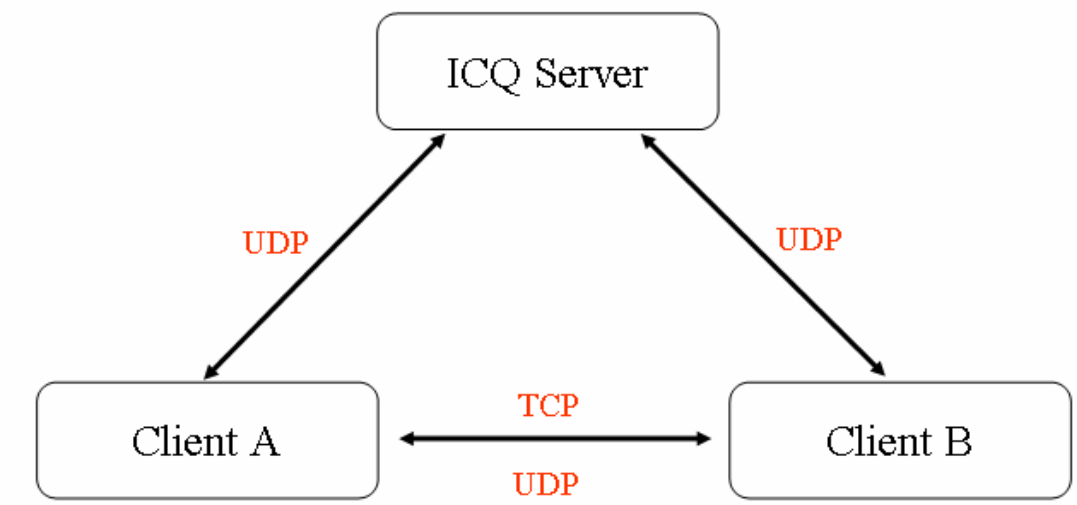

圖 2-7　ICQ 系統架構

主要的通訊方法有兩類:Client-Server 間通訊及 Client-Client 間通訊。

一、 Client-Server

Client 與 Server 間的通訊協定為 UDP，每個 UDP 封包加密後從 Client 端送到 Server，而 Server 回傳的訊息則不作加密動作。由於是用 UDP 在傳輸封包，又要能確保封包能送達，因此需要實作 reliable 機制，一端接收到封包之後，需回傳 ACK 回去。另外 ICQ 封包需加入自己的 Header，包含了 SEQ_NUM、COMMAND、SESSION_ID 等欄位，以便 Server 辨別封包是由哪個使用者所送出來的。

二、 Client-Client

Client 與 Client 間的通訊協定主要為 TCP，系統在使用者上線的時候，會告知其好友的狀態、IP、Port 等資料，當使用者需要傳送訊息時，便會先以 UDP 傳輸自己的 IP 及所開啟的 port 給對方，對方再根據資料連線過來建立 TCP 連線，之後便用 TCP 傳輸，進行聊天的動作，當使用者無法直接連線的時候，所要傳送的訊息便轉由 Server 轉送。

ICQ 系統架構的優點在於：高效率及伺服器不需要額外佔用連線數目。但其缺點有：需實作偵測斷線機制、程式開發的成本相當高、訊息易被攔截攻擊。

2.3.2 AOL

AOL 的系統構造圖如圖 2-8。

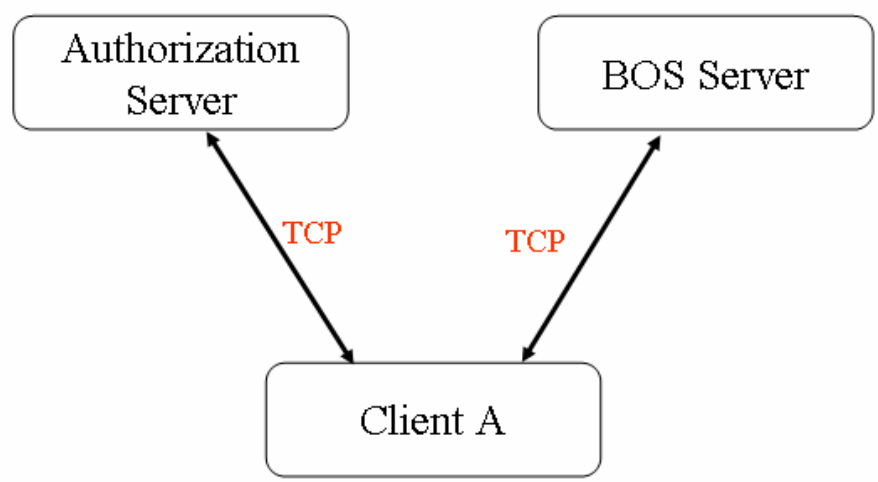

圖 2-8　AOL 系統架構

Oscar(Open System for Communication in Realtime)是 AOL 所發展出來的 IM 系統通訊協定。AOL 是多 Server 架構，主要分為兩種 Server：Authorization Server 及 BOS (Basic Oscar Service) Server。Authorization Server 負責認證的動作，以及回傳給使用者 Cookie，以便使用者連線至 BOS Server 時確認身份。BOS Server 主要是提供線上狀態及即時訊息服務。Server 與 Client 間的通訊是以 TCP 來傳輸。

系統運作的流程主要分為兩個大步驟：

一、 連線至 Authorization Server

連線至 Authorization Server，Server 會回傳 Connection Acknowledge 的訊息。使用者再發出 Authorization Request，Server 所回傳的 Authorization Response 中會帶有 Cookie 以及 Client 所要連線的 BOS Server 位置，此時 BOS Server 由 Cookie 便可辨認身份，不需要再作認證的動作。

二、 連線至 Authorization Server 所建議的 BOS Server

連線至 Authorization Server 所建議的 BOS Server ，Server 會回傳 Connection Acknowledge 的訊息。使用者再送出 BOS Sign On 的指令，Server 回傳 BOS Host-Ready 訊息告訴使用者已經準備好能開始服務。使用者詢問 Server 他所能傳輸的最大速率為何，當超過此值時，連線將自動中斷。使用者向 Server 要求好友的資訊，Server 回傳便將資訊儲存下來。使用者送出 Client Ready 的指令，告知 Server 已經準備好能開始服務。

AOL 架構的優點在於：無須實作偵測斷線機制、減少被攻擊的可能性。但其缺點是 BOS Server 間的通訊量極大，人數多時會影響其可延展性。

2.3.3 MSN Messenger

MSN Messenger 是微軟公司在 Windows 平台上所提供的，在 linux 上也有實作一套功能較少的版本，也由於程式不需再另外安裝，故使用者人數增加十分快速，2000 至 2001 的使用者人數就由九百萬人增加到一千八百萬人。

MSN Messenger 所提供的功能如下：

- 傳送訊息
- 支援多國語言
- 傳送檔案

- 影音交談
- 邀請好友玩遊戲
- 封鎖使用者訊息
- 遠端協助
- 共用白板

比前面所提到的線上即時系統功能更多，因此要支援如此多的能，便需要更有效能的系統。MSN Messenger 的系統構造如圖 2-9。

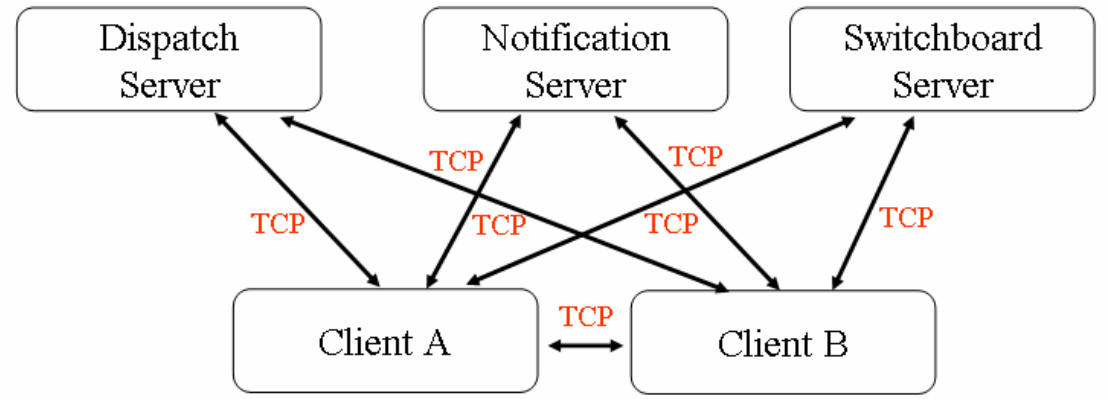

圖 2-9　MSN 系統架構

由於 MSN Messenger 有網站公布其通訊協定，因此我們可以知道更確切的資訊，它彼此間都是用 TCP 連線來傳送訊息，主要的通訊都是在 Client 與 Server 間，只有部分功能會使用 Client 與 Client 間互連，如檔案傳輸。

它主要分成三種 Server：Dispatch Server、Notification Server、Switchboard Server。Dispatch Server 負責認證及分配使用者到適當 Notification Server。Notification Server 負責線上狀態服務，記錄使用者的線上狀態及所在位置等資訊。Switchboard Server 負責即時訊息服務，轉送使用者的聊天訊息，與其他 Server 間的通訊量極少。

MSN Messenger 運作的步驟如下：

1. 使用者連線至 Dispatch Server。
2. Dispatch Server 回傳一台可用的 Notification Servers 給使用者。
3. 使用者連線至 Dispatch Server 所分配的 Notification Server，並取得好友的線上狀態及位置等資訊。
4. 當使用者狀態改變或是要與好友聊天時，會發出 Request 告知 Notification Server。
5. 若使用者要與好友聊天，Notification 會回傳一適當的 Switchboard Server，並告知對方的 Notification Server 也通知對方需連線至此 Switchboard Server。
6. 使用者便透過 Switchboard Server 傳送訊息。

另外我們還對 MSN Messenger 的 log 做分析，得到了兩點發現：當一使用者短時間重複登入，所分配到的 Notification Server 都不相同，可知 MSN 並未考量 Locality 的性質，並沒有將彼此有好友關係的人集中起來。另外在分配 Switchboard Server 的部分同樣也是會被分配到不同的 Server，因此聊天視窗一增加，連線數也會跟著增加。

原則上，MSN 架構保有 AOL 架構的優點，此外 Switchboard Server 影響增加可延展性。

綜合以上所介紹的三種著名線上即時傳呼系統，我們的設計凸以考慮以下幾點：

1. Client 間直接建立連線容易有網路安全上的問題，除多媒體的訊息之外，所以我們只考慮用 Client 與 Server 通訊的情形。
2. Server 之間實際如何溝通，從外界訊息並無法得知，所以我們需要針對可能的因素來進行比較。
3. MSN Messenger 中 Switchboard Server 的設計，也會納入我們的實驗中。

第三章 線上即時傳呼系統的設計

在這一章中，主要介紹我們所設計的線上即時傳呼系統的架構第 3.1 節介紹我的設計目標，第 3.2 節介紹我的設計內容，第 3.3 節說明 Login Server 的設計，第 3.4 節說明使用者資訊的維護方式，第 3.5 節介紹即時訊息服務的運作方式，第 3.6 節介紹伺服器之間的通訊方式。

3.1 設計目標

希望能讓系統盡可能支援越多使用者越好。因此，實驗主要分成兩步驟：

1. 找出單一伺服器的極限，能支援到多少人。
2. 測試多種架構，得出多台伺服器架構的極限，並盡可能的提升最大支援人數。

3.2 設計

多台伺服器主要需要考量的因素如下：

1. 延展性

最大連線數、記憶體、CPU 及網路傳輸都會是單一伺服器的問題，因此需要增加伺服器。

2. 平衡伺服器的 CPU 負載量，可增加延展性。

3. 容錯

單一伺服器容易因為被攻擊就造成整個系統當機，增加多台伺服器可避免此情形。

而多台伺服器對於架構上的差異，經分析之後有四種考量因素：

一、依照功能來分離伺服器

1. Login Server(LS)負責認證及分配使用者。
2. Messaging Server(MS)負責線上狀態服務。
3. Talk Server(TS)負責即時訊息服務。

二、維護使用者資訊

1. 集中型維護，由一台伺服器來維護所有使用者的資訊。
2. 分散型維護，由多台伺服器利用廣播的方式來維護使用者資訊。

三、即時訊息服務的運作方式

1. 由傳送訊息的使用者所在的伺服器轉送訊息給接收訊息的使用者所在的伺服器，再轉送給接收訊息的使用者。
2. 使用者雙方共同連線至另外的聊天專用伺服器進行聊天。
3. 傳送訊息的使用者額外建立連線至接收訊息的使用者所在的伺服器進行聊天。

四、伺服器之間的通訊方法

1. 伺服器之間是使用 TCP 或是 UDP 來溝通。
2. 伺服器之間是否使用額外的內部區域網路來連接。

3.3 Login Server

若只是單純的將伺服器加多，必定會出現問題，例如在登入的時候，使用者並不知道該連線到哪台伺服器，而且也需考量到伺服器負載的平衡性。

這個問題的解決方法需要一台 Login Server，來分配使用者到適當的伺服器。如圖 3-1，A 先連線到 Login Server，接收到要連線到 MS1 的訊息，便連線到 MS1。

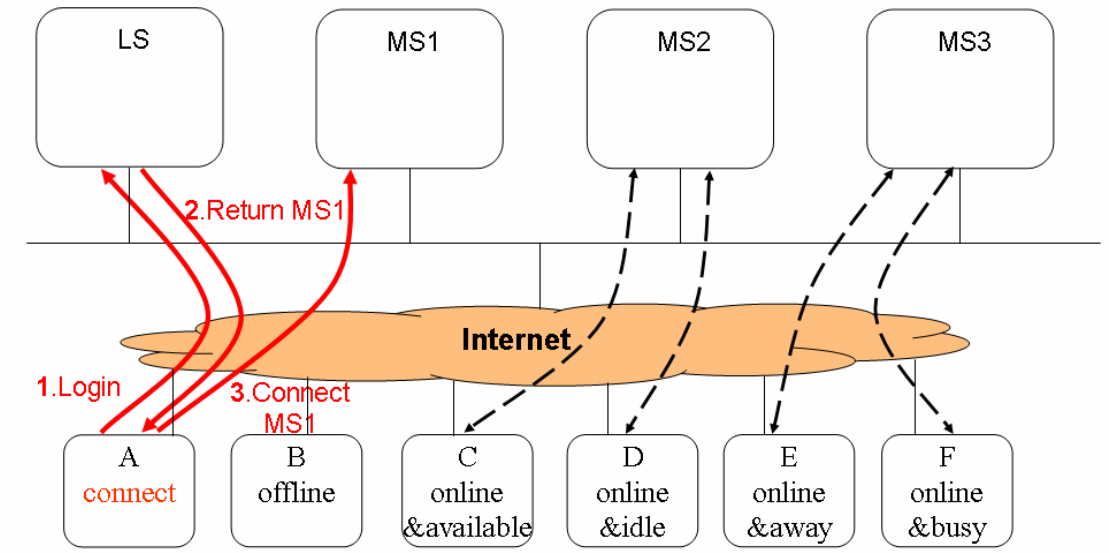

圖 3-1 Login Server 分配使用者

3.4 維護使用者資訊

在傳送訊息的時候，伺服器並不知道其他伺服器上面有哪些使用者，當被傳訊的對象不在此伺服器時，便不知道如何轉送訊息。如圖 3-2，使用者 A 要送訊息給使用者 E，但是 MS1 並不知道 E 位在 MS3，便無法轉送訊息。

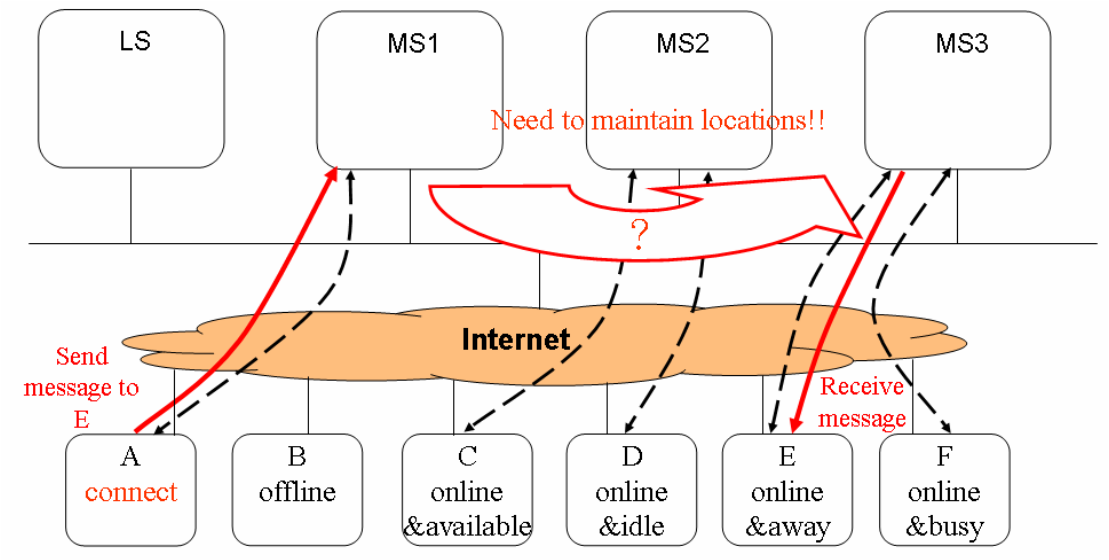

圖 3-2 多伺服器架構轉送訊息的問題

因此，伺服器需要額外維護一張使用者所在位置的清單，而為了保持各個伺服器上清單的一致性，可分為兩種方法，集中型及分散型，以下對於這兩個方法作詳細的介紹。

一、 集中型維護：

這個方法是說由一台伺服器來記錄線上所有使用者位置的清單，當各個伺服器需要去取得位置的時候，便來問這台伺服器，在我們的實驗中，我們讓 Login Server 擔任這個角色，負責記錄線上所有使用者的位置。

如圖 3-3 所示，Login Server 維護一張線上所有使用者的清單，而其他伺服器只需記錄在此台伺服器上的使用者的好友資訊。

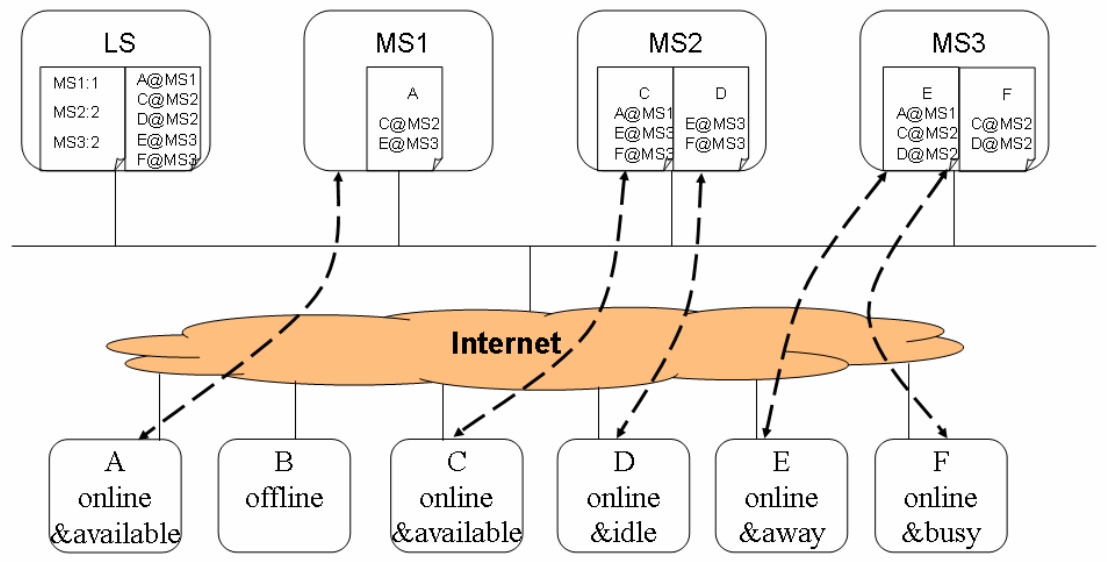

圖 3-3 集中型清單維護的架構

接著我們對於第二章提到的四種使用者行為來介紹此種架構的運作方式。

1. 登入：

登入動作可以分為兩個階段，第一階段是與 Login Server 溝通，取得所要連線的伺服器資訊，Login Server 傳給使用者 A 要連線到 MS1。第二階段是使用者與伺服器建立連線之後，伺服器需要通知 Login Server 去修改清單，並且取得使用者的好友資訊，再傳送給各個好友此使用者的上線訊息。如圖 3-4，MS1 先通知 Login Server 使用者 A 已經上線，需要修改線上所有使用者位置的清單，接著 Login Server 會回傳 A 的好友資訊，告訴 M1，A 的好友 C 在 MS2，E 在 MS3， MS1 再通知 MS2、MS3，MS2 會將 A 的上線資訊通知 C，MS3 會將 A 的上線資訊通知 E。

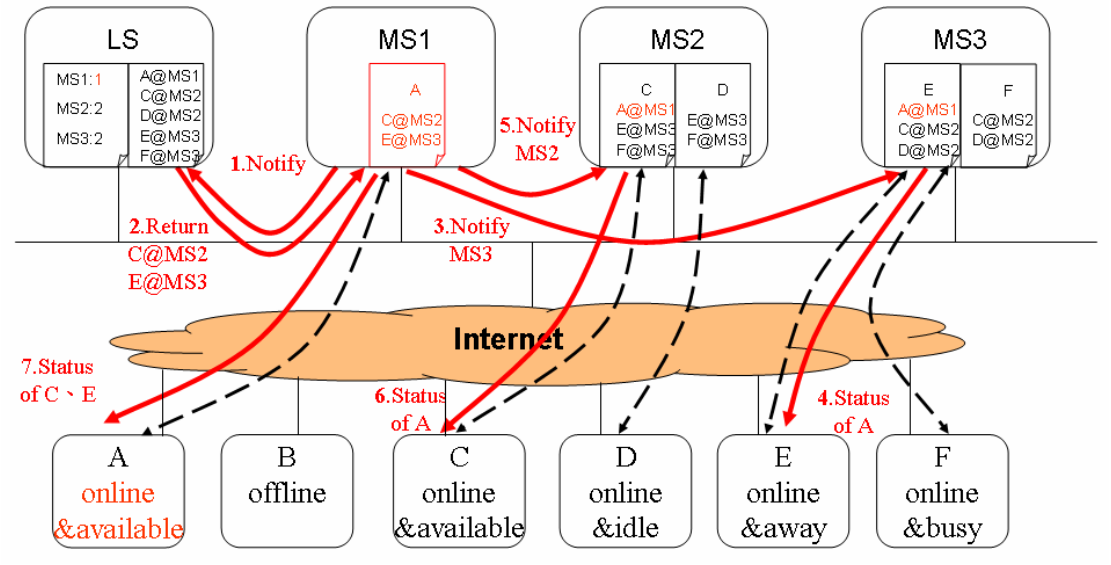

圖 3-4 集中型清單維護-使用者登入

2. 登出：

當伺服器接收到使用者登出的訊息時，會先通知 Login Server 修改清單，再通知好友所在的伺服器，由他們通知好友離線的訊息。如圖 3-5，A 送出要登出的訊息，MS1 先通知 Login Server 修改使用者清單，再通知 C、E 所在的 MS2、MS3，由他們通知 C、E 使用者 A 登出的訊息。

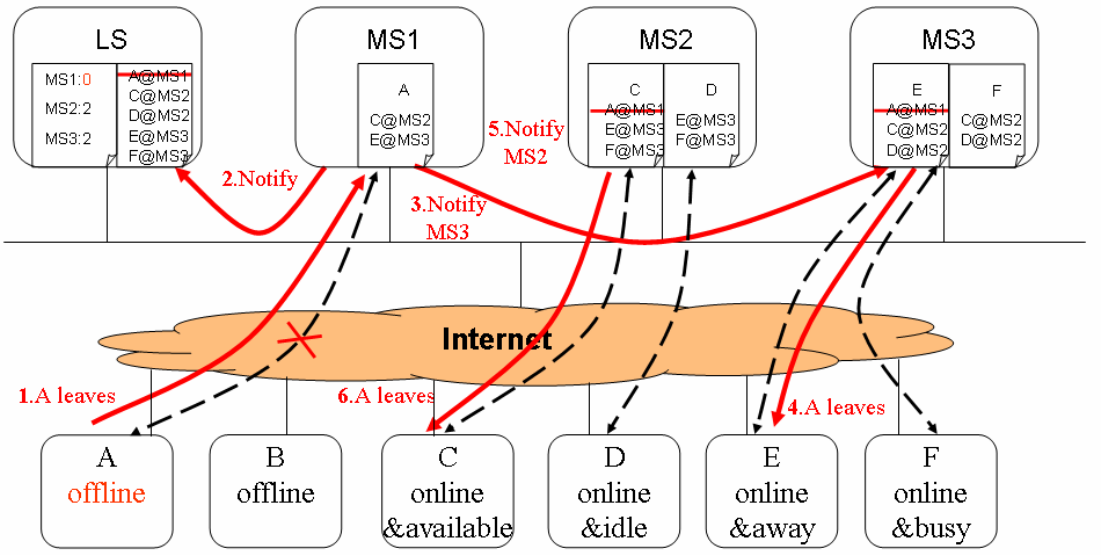

圖 3-5 集中型清單維護-使用者登出

3. 傳送訊息：

當使用者之間要互相傳送訊息的時候，伺服器可以直接查詢自己所維護的好友位置清單，轉送訊息到該伺服器，再傳給好友。如圖 3-6，A 要傳送訊息給 E，MS1 接收到訊息時，查詢好友位置清單得知 E 在 MS3，便將訊息轉送給 MS3，再由 MS3 傳給 E。

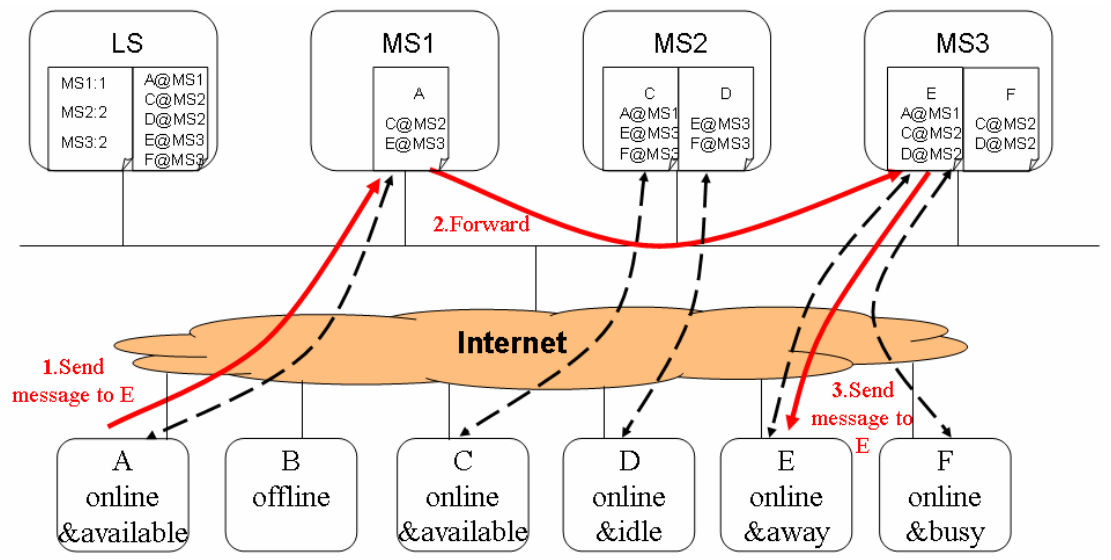

圖 3-6 傳送訊息

4. 改變狀態：

當伺服器接收到使用者改變狀態的訊息時，需要將訊息轉送給好友。運作方式跟登出的通知方式大致相同。

二、 分散型：

在這個方法中，我們利用廣播訊息的方式來維護各個伺服器清單的一致性，當需要修改清單資訊的時候，伺服器便發出廣播訊息通知所有伺服器，去修改清單。

如圖 3-7，Login Server 不需要額外維護線上所有使用者的清單，由各個伺服器去進行廣播的動作，達到清單的一致性。

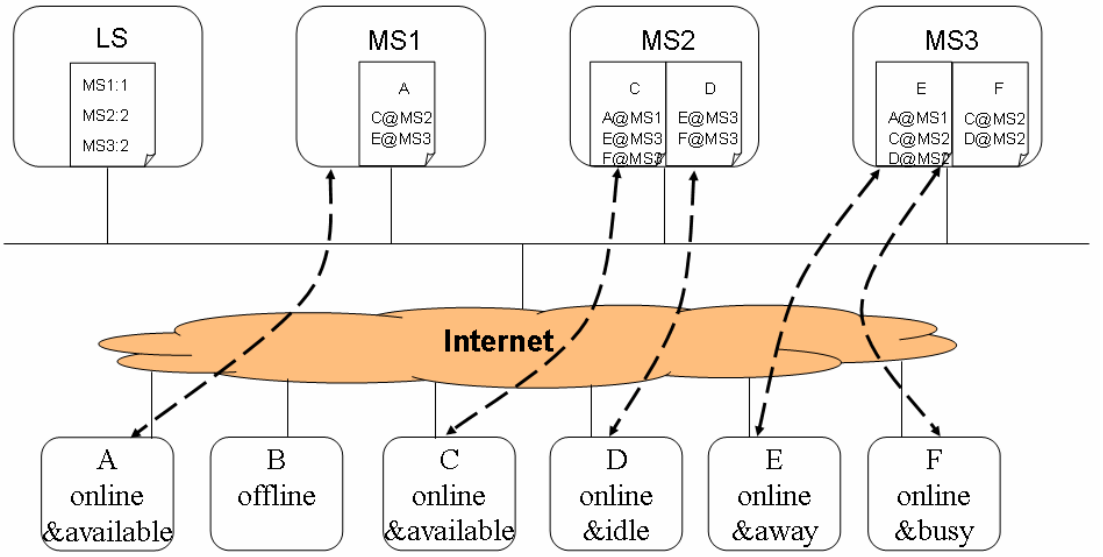

圖 3-7 分散型清單維護的架構

同樣的，我們由四種使用者的行為來解釋此種架構的運作情形。

1. 登入：

在登入部份同樣分為兩個階段，第一階段是與 Login Server 聯繫取得所要連線的伺服器，第二階段伺服器要發出廣播訊息，告訴所有伺服器此使用者上線的訊息，以便修改清單，當其他伺服器收到廣播訊息，便會回傳位在此伺服器上的好友資訊。如圖 3-8，MS1 發出廣播訊息告訴其他伺服器 A 上線的訊息，MS2 接到訊息，便回傳 C 的資訊，並通知 C，A 已上線，MS3 接到訊息，便回傳 E 的資訊，並通知 E，A 已上線。

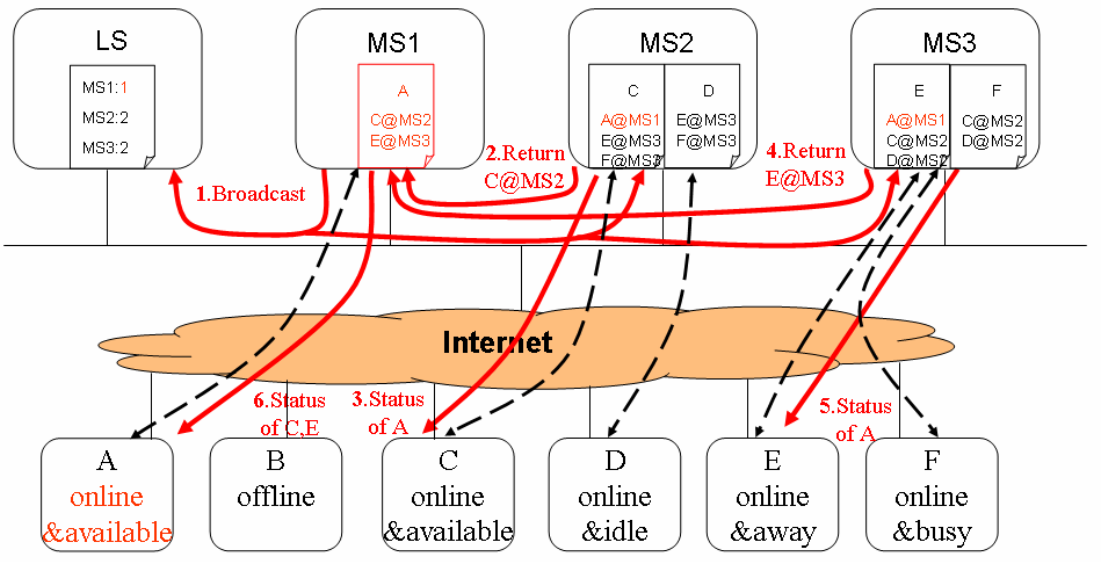

圖 3-8 分散型清單維護-使用者登入

2. 登出：

當伺服器接到使用者要登出的訊息時，便發出廣播訊息通知其他伺服器，修改清單，並且通知好友。如圖 3-9，A 送出登出訊息，MS1 便廣播告訴其他伺服器該修改清單，而 MS2 接到廣播訊息，便通知 C，A 已離線，MS3 接到廣播訊息便通知 E，A 已離線。

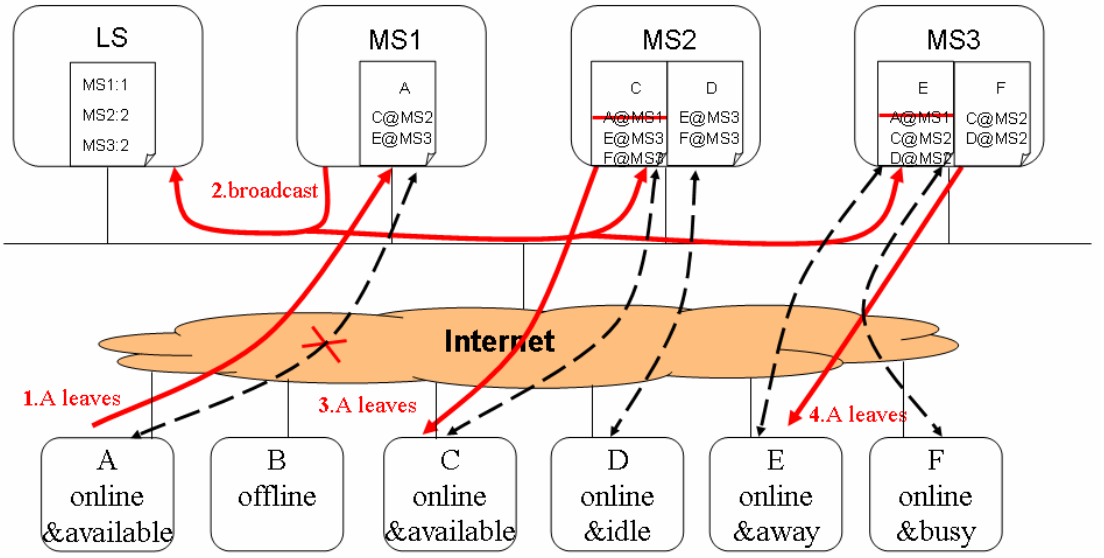

圖 3-9 分散型清單維護-使用者登出

3. 傳送訊息：

這部分的運作方式與集中型的方式相同，由伺服器負責轉送訊息給其他伺服器的好友。

4. 改變狀態：

當伺服器接收到使用者改變狀態的訊息時，便會利用清單查出好友的位置，並進行轉送的動作，由好友所在的伺服器再通知好友。與登出的運作情形一樣。

3.5 即時訊息服務的運作方式

對於四種使用者行為，對伺服器負擔最大的便是傳送訊息，對於即時訊息服務的處理方式便會影響系統所能負擔的人數上限，若是能把這功能獨立出來，由聊天專用伺服器(Talk Server)來服務，便可以減少伺服器之間的通訊量，可以增加系統的效能。如圖 3-10 所示，當 A 要與 E 互相傳送訊息的時候，MS1 會先詢問 Login Server 該連線到哪個傳送訊息專用伺服器，Login Server 回傳要連線至 TS1，MS1 便通知 A 要連線到 TS1，並且轉送給 MS3，由 MS3 通知 E 要連線到 TS1。

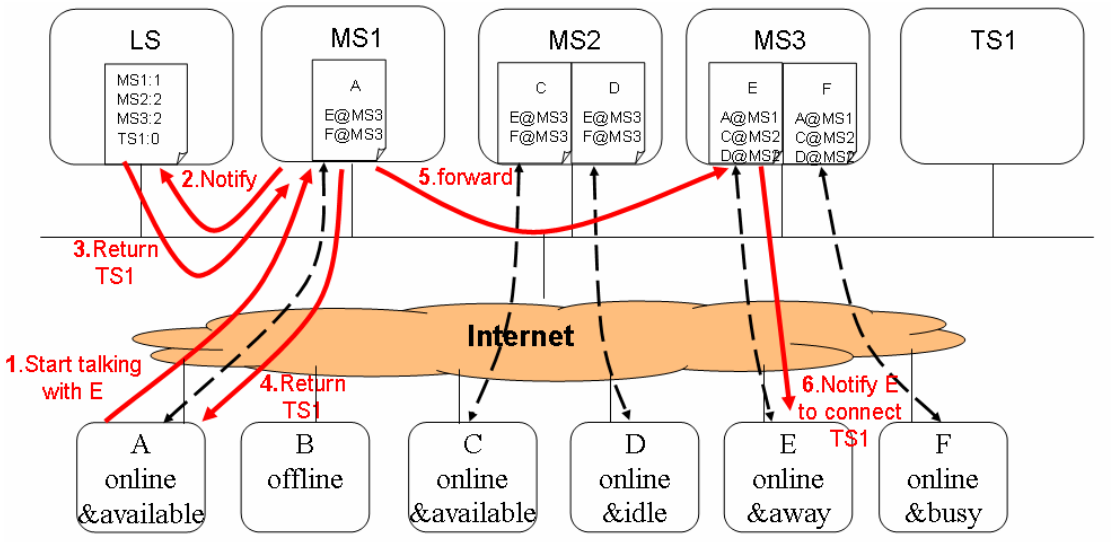

圖 3-10 取得聊天專用伺服器的資訊

接著 A 及 E 便另外與 TS1 建立連線，如圖 3-11， TS1 只需通知 Login Server 去修改人數的清單，之後 A 便可以透過 TS1 與 E 溝通，如同單一伺服器的轉送方式。

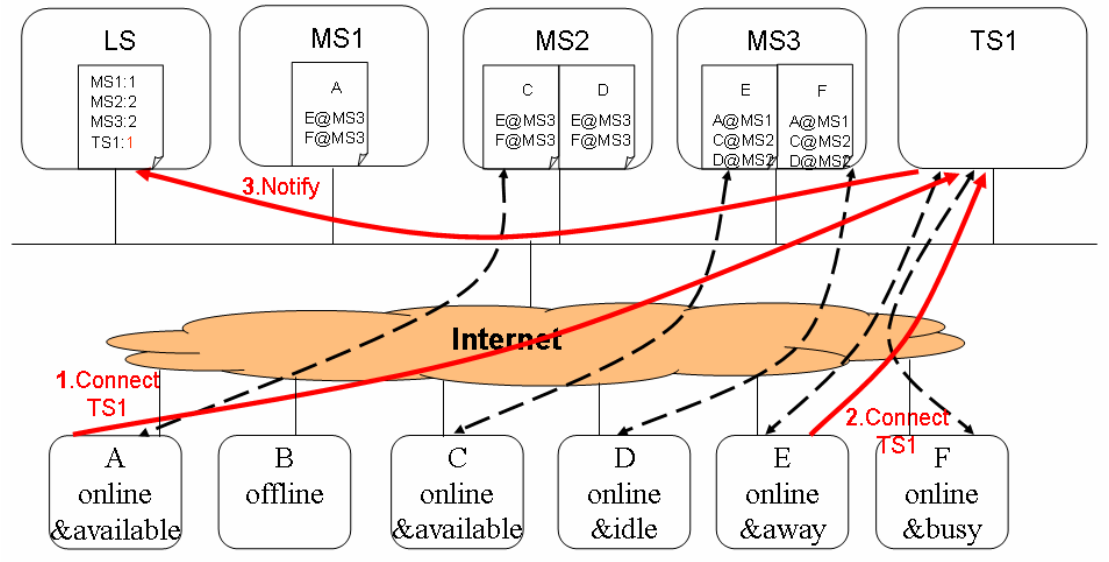

圖 3-11 連線至聊天專用伺服器

這個想法是來自於 MSN Messenger 的架構，將 Talk Server 獨立出來，以減少 Server 間通訊，將工作簡單化以提升效能，但是也可能因為分離這動作使得原本只需要少量伺服器就可以服務的工作量，卻得用更多伺服器來運作。因此我們還考量了另外一種作法(Modified Talk)，如圖 3-12，當 A 告訴 MS1 要與 E 傳送訊息時，MS1 便會回傳 MS3 的位置叫 A 另外連線過去，如此 A、E 便都會在 MS3 上，透過 MS3 轉送訊息。這樣的作法對於伺服器的 loading 也較平均，將在之後對於這幾種作法做實驗比較。

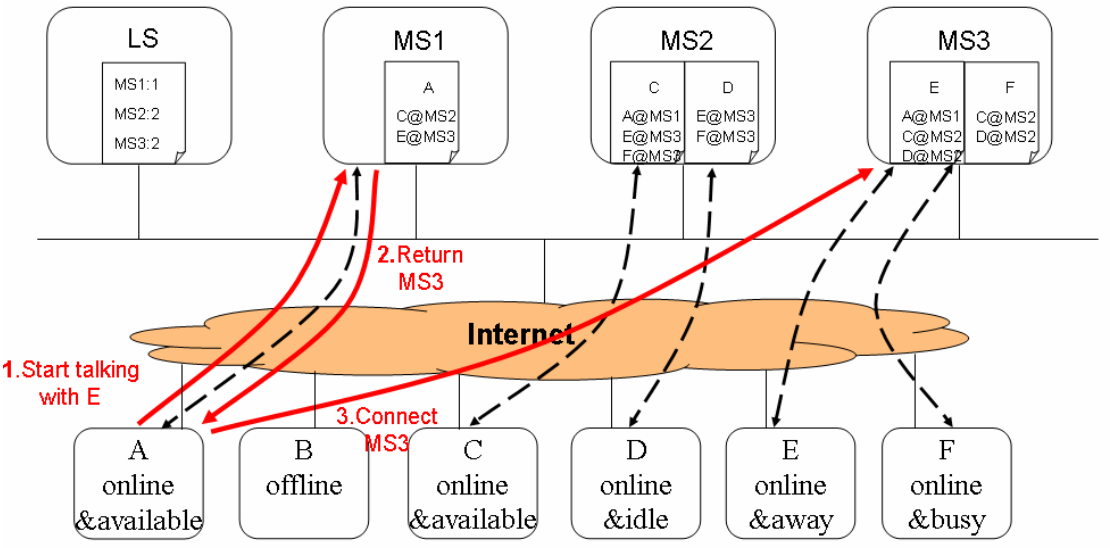

圖 3-12 改良 Talk Server

3.6 伺服器間的通訊方式

一、TCP 或 UDP

前面所提到的分散型架構中，伺服器使用了廣播的方法來維護清單的一致性，但是如果伺服器個數一變多，以 TCP 來廣播需要對每一條連線都傳送訊息，便會顯得較沒有效率，如果能以 UDP 來進行廣播，便可以減少伺服器所傳送的訊息，以增進效能。

二、是否使用額外的內部區域網路

前面所提到的多伺服器架構運作方式，伺服器之間的通訊，都是經由對外的區域網路來做溝通，這樣伺服器間通訊的封包必定會跟伺服器與使用者間的通訊衝突，如圖 3-13，若能架設額外的內部區域網路，來分散這兩種通訊量，減少封包衝突，減少重傳便可以提升系統的效能。另外 UDP 的部分，考慮封包遺失及衝突的可能性，必定要使用額外的內部區域網路才行。

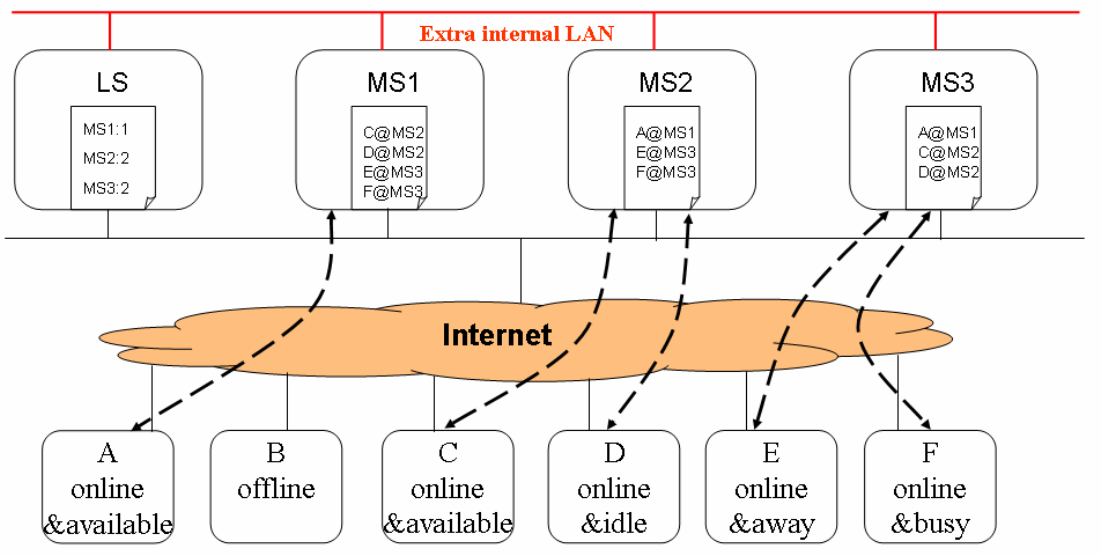

圖 3-13 架設一個額外的內部區域網路

第四章　實作分析

在這一章我們對於第三章所介紹的架構進行實驗，進而比較何種架構效能較好。

4.1 實驗內容

第三章介紹了許多架構，集中型清單維護與分散型清單維護，使用 TCP 或 UDP，是否架設額外內部區域網路，好友集中等等性質，由於彼此並不完全衝突，能夠個別實驗該性質做出比較，關於即時訊息服務的運作方式是我們較感興趣的部分，因此先得出其他性質中較好的架構，再進行這部分的實驗，以下先做個小整理，列出我們所要做的實驗。

一、　集中型維護使用者資訊

1. 使用 TCP
   a. 使用額外內部網路　(CM-TCP-XL)
   b. 不使用額外內部網路　(CM-TCP-NL)
2. 使用 UDP
   a. 使用額外內部網路　(CM-UDP-XL)
   b. 不使用額外內部網路，在與外界封包互相衝突的情形下，packet loss 會很高，較沒效率，故不考慮此情形。

二、　分散型維護使用者資訊

1. 使用 TCP，達到 broadcast 的功能，需一個一個傳送，較沒效率，故不考慮此情形。
2. 使用 UDP
   a. 使用額外內部網路　(DM-UDP-XL)

b. 不使用額外內部網路,在與外界封包互相衝突的情形下,packet loss 會很高，較沒效率，故不考慮此情形。

除了上述所列的四項實驗，另外對於即時訊息服務的運作方式像 Talk Server 的運用以及前面曾提到的改良 Talk Server 方法(Modified Talk)，聊天前先主動連線至對方伺服器，加以實驗比較。

4.2 實驗步驟

1. 收集現在 IM 系統的訊息記錄，以便我們模擬。由於台灣使用 AOL 的使用者並不多，在這裡我們收集 MSN Messenger 的使用者訊息記錄。
2. 利用所取得的訊息記錄，測試單一伺服器架構下，找出所能服務的最大人數有多少。
3. 利用 2 所得到的結果來衡量我們所需要的資源，找出多伺服器架構下，所能服務的最大人數有多少。

4.3 收集線上即時傳呼系統使用者的訊息記錄

由圖 2-7 可知 AOL、MSN Messenger 是使用者人數較多的兩種 IM 系統，由於台灣地區使用 AOL 的使用者並不多，因此我們決定要收集 MSN Messenger 的使用者訊息記錄，在 gateway 上利用 tcpdump 這個指令取得 MSN Messenger 的封包，收集對象是一個有 20 人的團體，我們收集一個月的資料，從當中取得 30 分鐘內訊息傳輸量最大的那段資料來進行實驗。圖 4-1 為我們所取得的訊息與時間分佈。

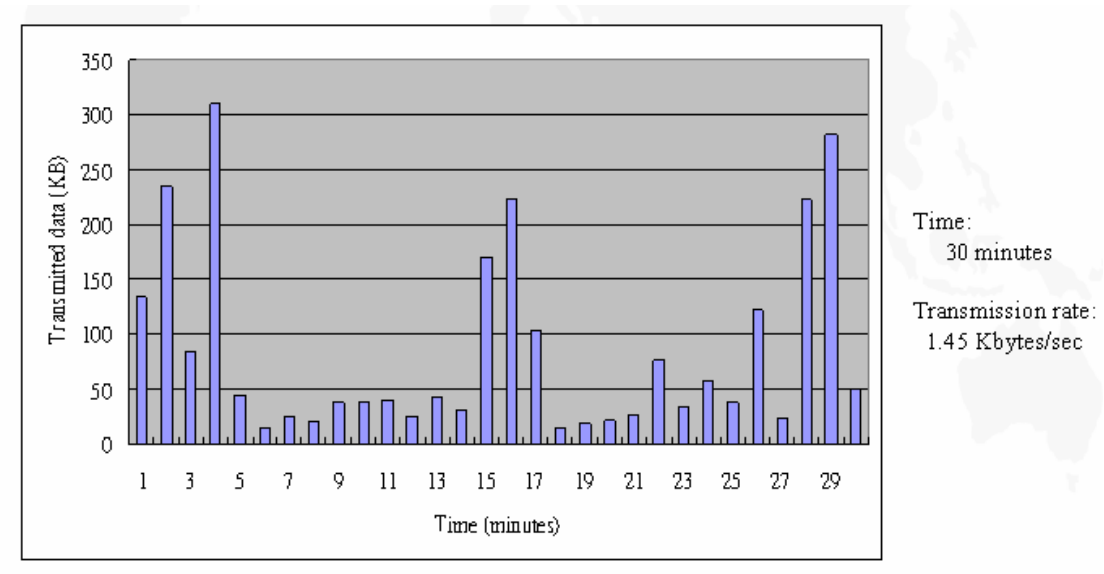

圖 4-1　30 分鐘的訊息分佈

如圖 4-2，在這段資料中，除了團體內的 20 個使用者之外，外界的好友還有 11 個人，而我們要用此資料來模擬的話，需要讓 31 個人都上線。

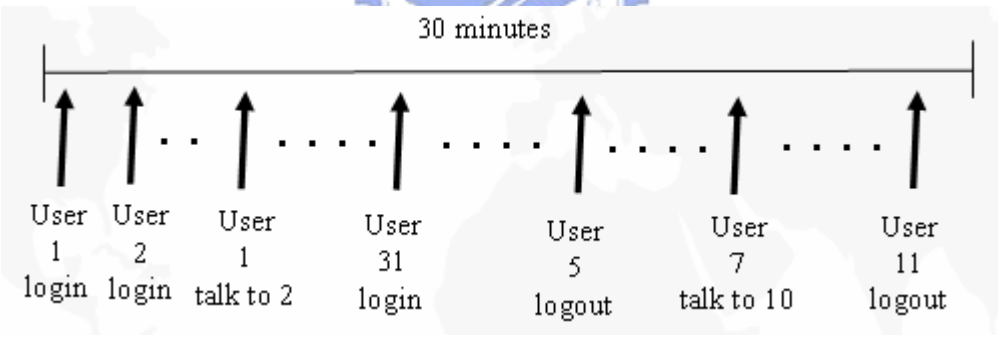

圖 4-2 測試資料

由於我們所取得的資料是當時網路流量最大的情形，所以我們直接將這段資料做複製的動作，如圖 4-3，來模擬線上即時傳呼系統短時間收到大量封包的情形。

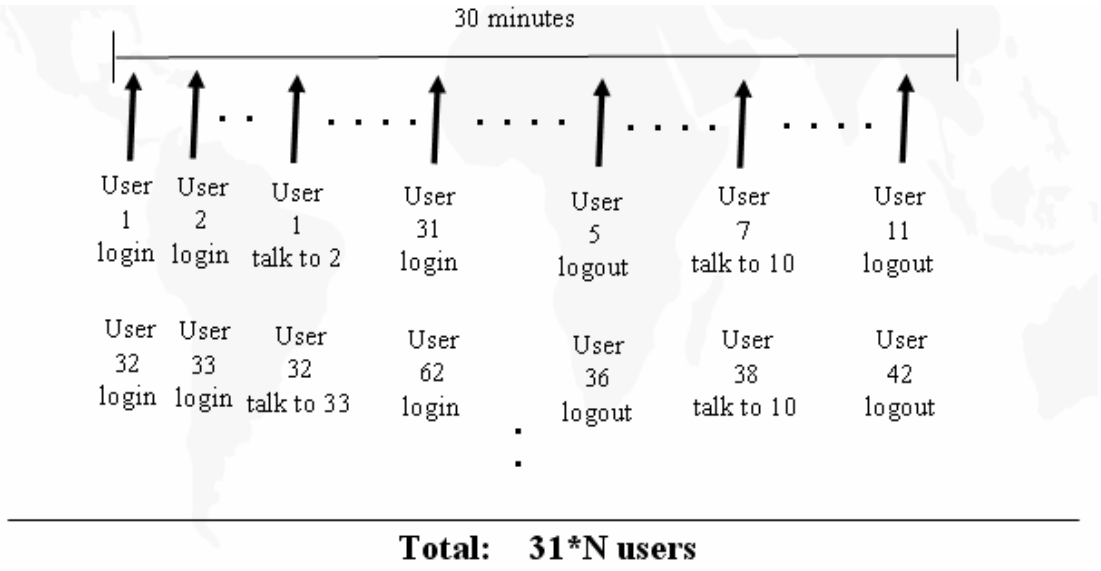

圖 4-3 複製測試資料

並且我們將這一個月的訊息做了統計，主要統計改變狀態、傳送聊天訊息，以及與 Talk Server 建立連線的訊息個數，如圖 4-4。

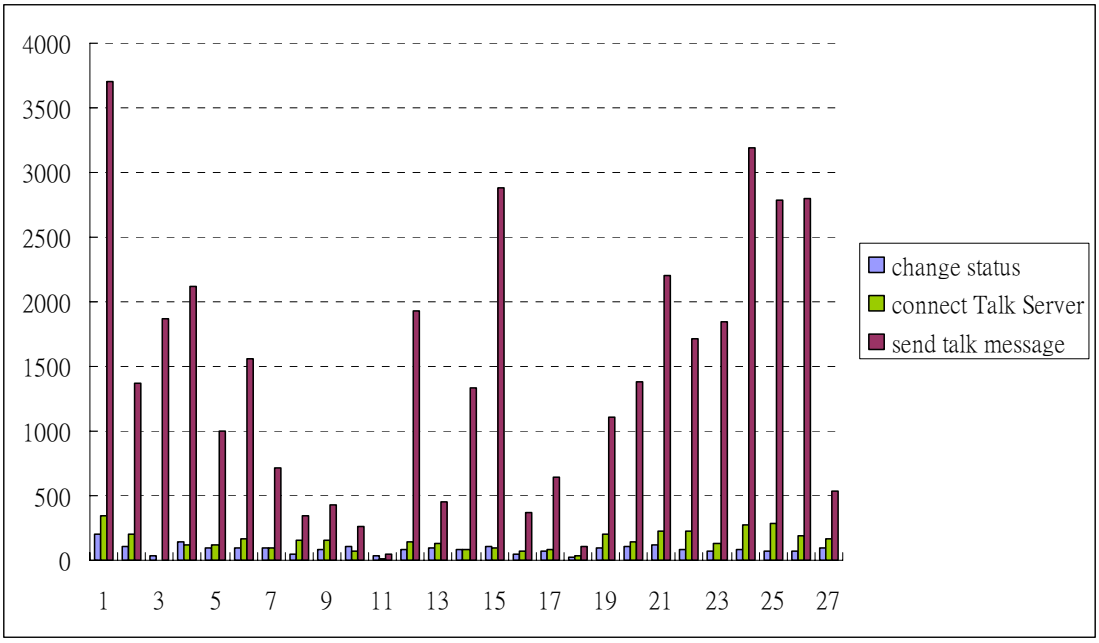

圖 4-4 一個月內的訊息分佈

我們可以看出傳送聊天訊息的次數遠高於其他的訊息，因此 Talk Server 所能帶來的效果，必定能減少非常龐大的資料量。

4.4 實作

有了測試資料之後，接下來便是實作系統，再進行測試，除了第三章所提到的方法需要實作外，我們還使用了一些新的技術及方法。我們使用 FreeBSD 中的 KQUEUE [2]來取代傳統的 select，因為線上即時傳呼系統需要服務大量使用者，對於同時處理大量連線，KQUEUE 的效能勝過 select，這兩個指令運作上最大的不同就是，select 在呼叫時會傳出需要被檢查的名單，再由 kernel 一個一個檢查是否有反應，而 KQUEUE 可利用 kevent 這個指令對 kernel 註冊有興趣的連線，此清單會記錄在 kenerl 中，當連線有反應的時候，kernel 便會直接對清單作修改，當 KQUEUE 要取得有反應的清單的時候，可以直接將清單傳送給它，這也就是 KQUEUE 之所以效能勝過 select 的地方。

另外以使用 UDP 廣播的部分，由於封包可能遺失，且遺失會造成清單不一致，因此我們必須加入一些簡單的機制保證伺服器一定收

得到封包，且不要造成伺服器太大的負擔，我們利用傳送緩衝區(send buffer)，來記錄我們的 UDP 封包，等待前一個封包接收到 ACK 之後，再傳出下一個封包，並且定時檢查是否逾時，是的話要進行重傳封包的動作，當緩衝區中的封包目的地相同時，我們會將它合併之後再送出，以減少傳送次數。

4.5 測試環境

	Server	Client
OS	FreeBSD 4.9-RELEASE	FreeBSD 4.9-RELEASE
CPU	AMD Athlon™ XP 2000+	AMD Athlon™ XP 2000+
Memory	768MB~1536MB	512MB
Network adapter	3Com 3c905B-TX Fast Etherlink XL	3Com 3c905C-TX Fast Etherlink XL

表 4-1 測試環境

我們一共使用七台電腦當伺服器，十二台電腦用來模擬使用者端。開始測試前，我們需要知道用來模擬使用者端的電腦，究竟能模擬多少人，不能讓模擬使用者的電腦負擔太高，電腦在模擬 60000 個使用者的時候，平均的 CPU 使用率為 37.9%，前 10%的 CPU 使用率為 79.68%，所以我們讓每台模擬使用者的電腦，最多模擬 60000 個使用者。

4.6 測試單一伺服器

由 0 人開始，每次增加 10000 人，取得平均回應時間及伺服器負載量(CPU loading)，另外我們還計算前 10%高的回應時間平均，來當作我們判定系統是否負擔過重的判斷因素，一般使用者所以忍受的

延遲在三秒以內都還好，所以當 10%高的回應時間平均超過三秒時，我們便說此系統超過負荷。

我們得到單一伺服器架構下，在服務 120000 個使用者時，前 10%高的回應時間平均為 3013.45ms，剛好超過三秒，由前面所定模擬使用者的電腦最多模擬 60000 人，在這個測試情形便需要兩台電腦來模擬使用者。

4.7 測試多伺服器

測試方法如同單一伺服器架構，逐漸增加人數，直到前 10%高的回應時間平均超過三秒。我們以固定伺服器個數來進行實驗，測試四台的情形，是否使用聊天專用伺服器來提供聊天服務的情形，會影響實際服務的伺服器個數，以下是我們實驗的結果。

1 Login Server + 3 IM Servers

Case	Max users (max 10% T ≈ 3s)
CM-TCP-NL	280000
CM-TCP-XL	300000
CM-UDP-XL	250000
DM-UDP-XL	270000

表 4-2 支援的最大人數

首先我們先比較以上四種情形，對於前 10%高的回應時間平均與使用者人數的分佈如圖 4-5。

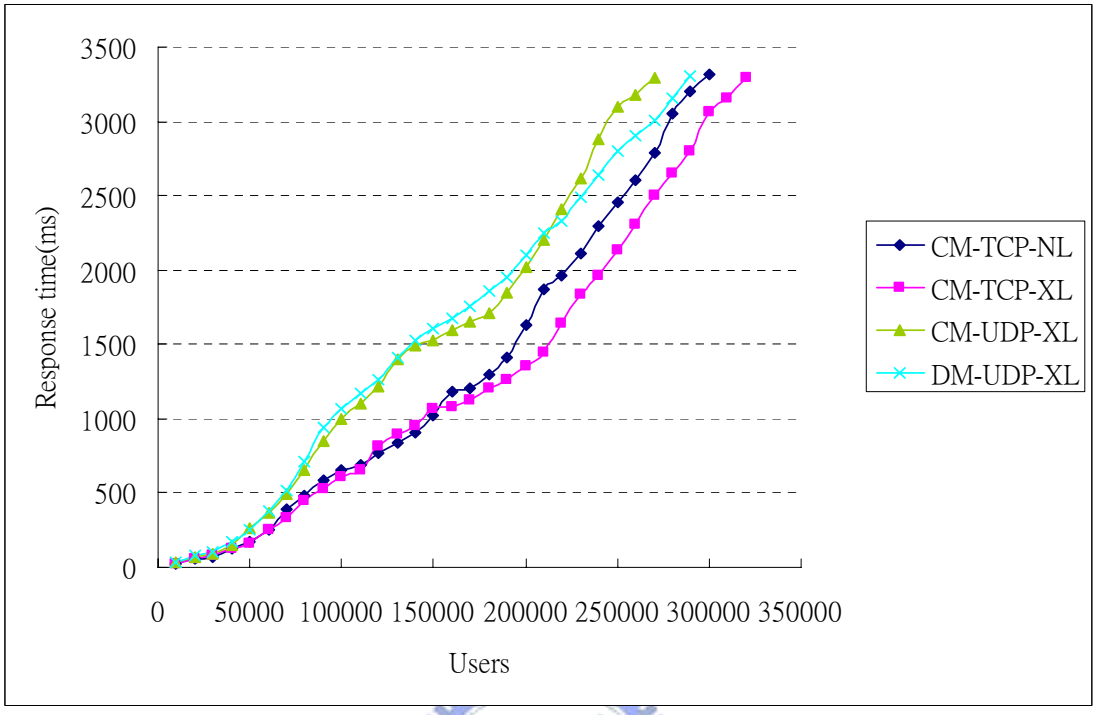

圖 4-5 前 10%高的平均回應時間與使用者人數關係圖

對於伺服器 CPU 使用量與使用者人數的關係圖如圖 4-6。

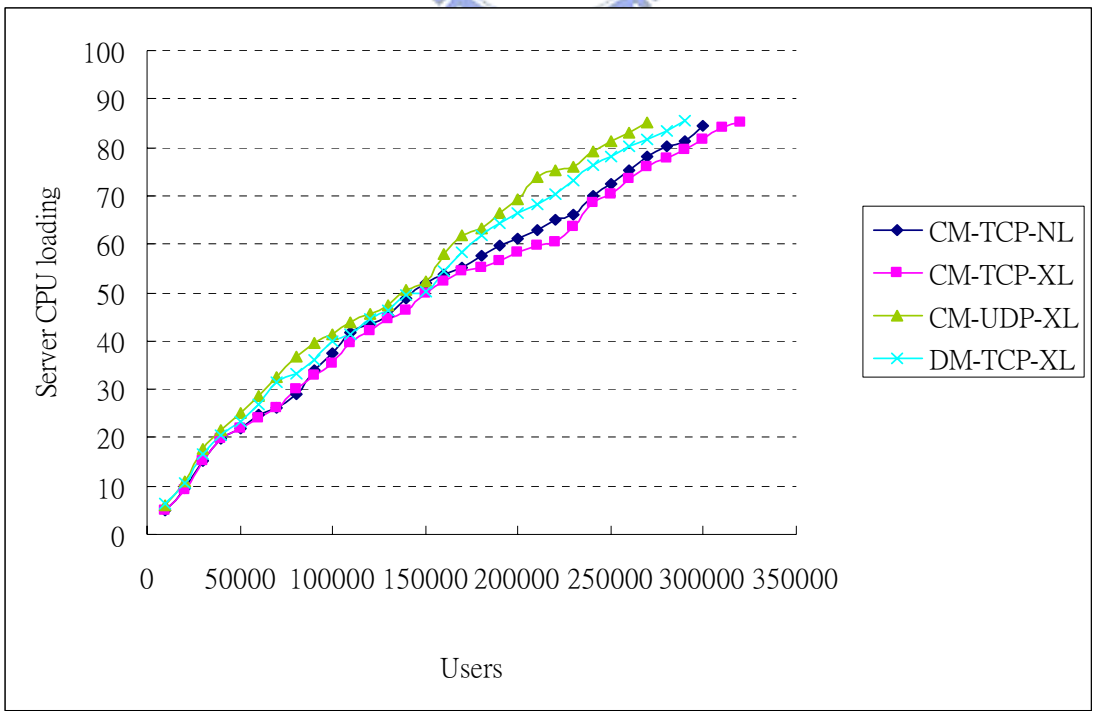

圖 4-6 伺服器 CPU 使用量與使用者人數關係圖

我們可以看出以 TCP 溝通的架設額外內部網路的集中型架構能支援的人數最多，我們原以為以 UDP 廣播的方法能減少訊息數量以增加效能，雖然 CPU 使用量並無顯著的增加，但是因為需要等待 ACK，反而降低了效能。

接著我們來看各自的比較情形，以單一伺服器所能服務的最大人數 120000 人的測試數據來當例子。首先我們先看有無架設額外內部區域網路的比較。

CM-TCP-NL 情形中前 10%的平均回應時間為 765.47ms，伺服器 CPU 使用率前 10%高的平均為 43.62%，CM-TCP-XL 情形中前 10%高的平均回應時間為 710.03ms，伺服器 CPU 使用率前 10%高的平均為 42.07%，這部分的結果很相近。

CM-TCP-NL 的收到封包數有 1851320 個，送出封包數有 1917451 個，而 CM-TCP-XL 接上了兩個區域網路，對外區域網路收到的封包數有 1745230 個，送出封包數有 1795617 個，而對內區域網路收到的封包數有 40175 個，送出的封包數有 39753 個，內外封包數量加起來比 CM-TCP-NL 少了不少封包，證明額外的內部區域網路有發揮作用。

接著來看集中型與分散型的比較。CM-UDP-XL 情形中前 10%高的平均回應時間為 1210.87ms，伺服器 CPU 使用率前 10%高為 45.57%，DM-UDP-XL 情形中前 10%高的平均回應時間為 1264.49ms，伺服器 CPU 使用率前 10%高為 44.59%，這部分的結果也很相近。

CM-UDP-XL 對內區域網路收到的封包有 52346 個，送出的封包數有 70292 個，而 DM-UDP-XL 對內區域網路收到的封包有 60357 個，送出的封包有 65035 個，由於分散型架構使用廣播的功能，所以伺服器接收的封包個數會較多，而送出的封包數會較少，由這個結果來看，可以得知分散型的架構效能較好。

接著來看 TCP 與 UDP 的比較。CM-TCP-XL 情形中前 10%高的平均回應時間為 710.01ms，伺服器 CPU 使用率前 10%高為 42.07%，CM-UDP-XL 情形中前 10%高的平均回應時間為 1210.87ms，伺服器 CPU 使用率前 10%高的平均為 45.57%，CPU 使用率的結果很相近，但是訊息回應時間卻大很多，這是因為 UDP 實作重傳及等待 ACK 的動作造成的，許多訊息會造成延遲送出，所以 UDP 在這裡並未達到原本預料的結果。

由前面的實驗結果可以知道集中型維護，使用 TCP 及額外內部區域網路的架構較佳，我們將以此環境再對即時訊息服務的運作方式，不使用 Talk Server(NT)、使用 Talk Server(T)、我們改良的方式(MT)，加以實驗比較。

圖 4-7 是三種方式在三台伺服器的環境下所得到的結果。Talk Server 的方法使用兩台 Talk Server。

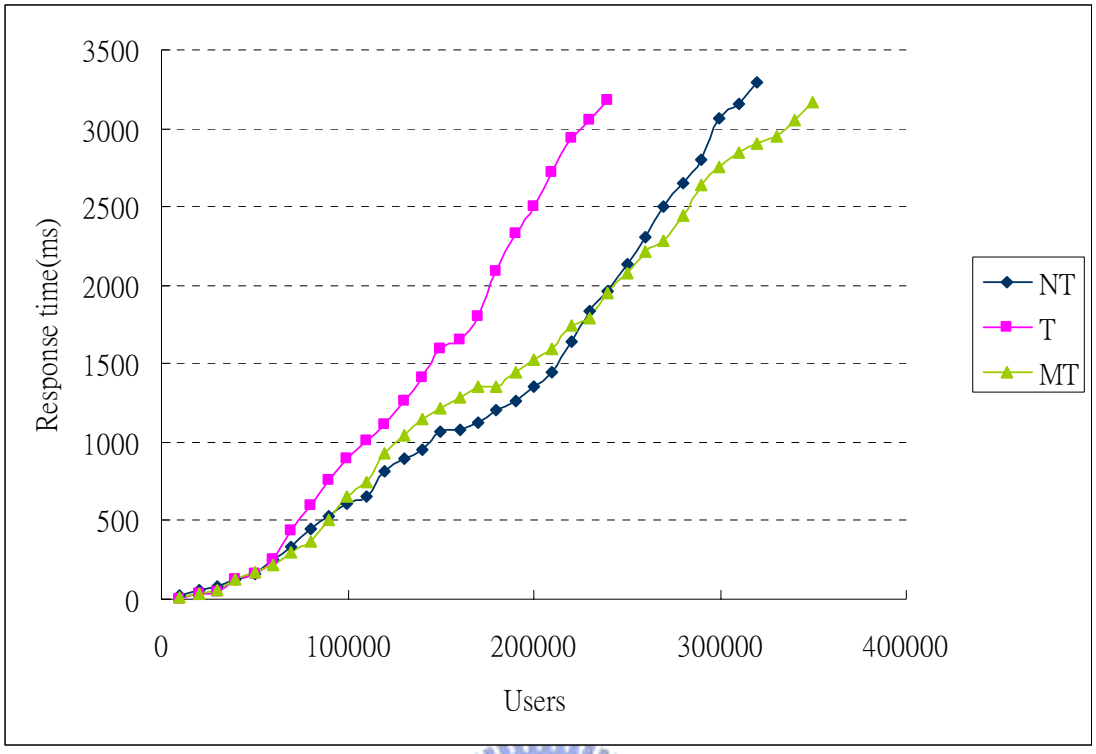

圖 4-7 三台伺服器時，即時訊息服務的運作方式的比較

由圖 4-7 可知，我們所改良的方法結果是最好的。為了實驗架構的 Scalibilty，我們將逐漸增加伺服器數目來進行實驗。以下是對四台、五台、六台伺服器進行實驗的結果。而使用 Talk Server 的方法分別使用三台、三台、四台 Talk Server 來進行實驗。圖 4-8 是三種方式在四台伺服器的環境下所得到的結果。圖 4-9 是三種方式在五台伺服器的環境下所得到的結果。圖 4-10 是三種方式在六台伺服器的環境下所得到的結果。

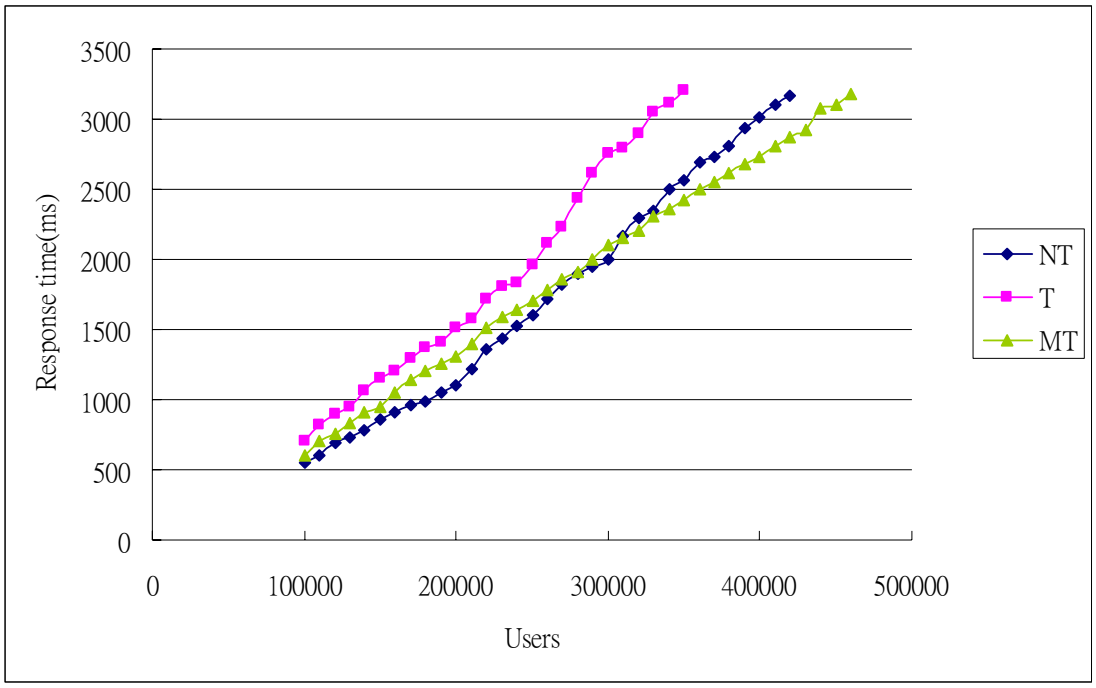

圖 4-8 四台伺服器時，即時訊息服務的運作方式的比較

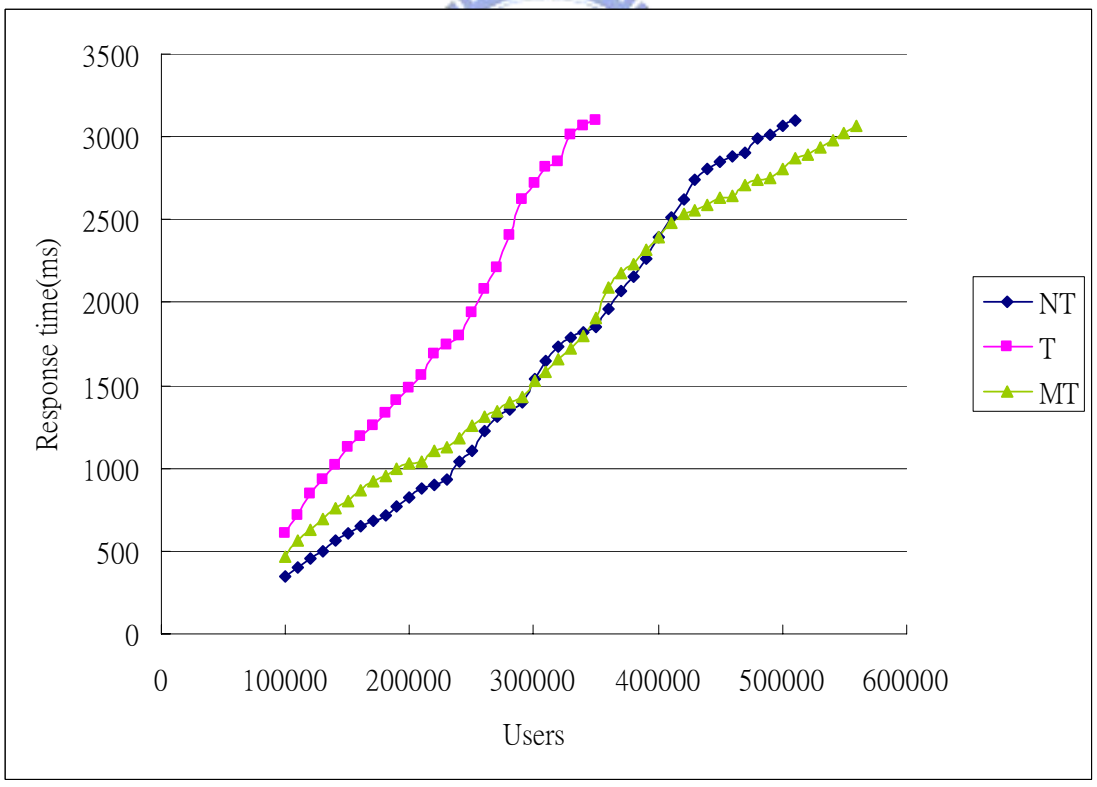

圖 4-9 五台伺服器時，即時訊息服務的運作方式的比較

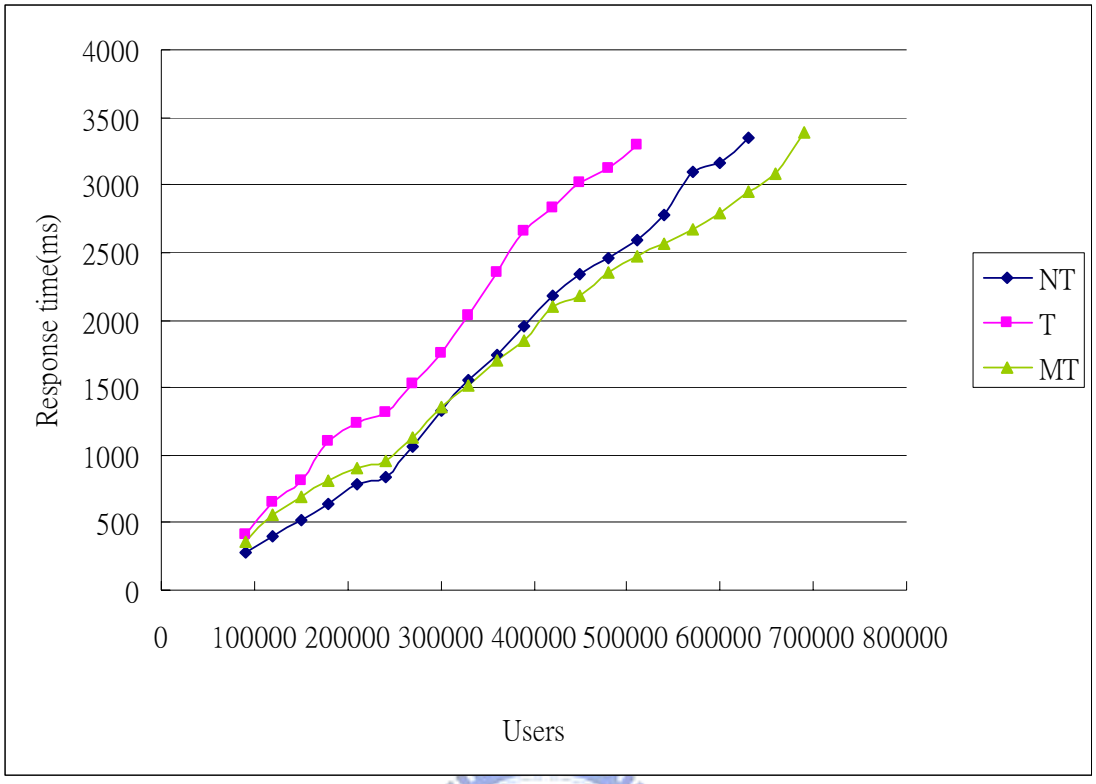

圖 4-10 六台伺服器時，即時訊息服務的運作方式的比較

由 4-7、4-8、4-9、4-10 可以看出，我們所改良的方法由於連線數會大量增多，所以在前期人數逐漸增加時稍差於 NT 的方法，不過當人數大到二三十萬以上時，MT 便會勝過 NT 的方法，而 T 的方法，主要限制在於 Talk Server 使用的台數，伺服器 loading 並不平均，而我們的方法仍然是最好的。

我們同時需確定網路狀況並無超過 Switch 所能承載的頻寬 100Mbit，因此還做了流量統計，將每種情況每秒最大的流量記錄下來，由於分成內外部區域網路，內部網路一定遠小於外部網路的流量，因此我們只分析外部網路的流量。以下圖 4-11、圖 4-12、圖 4-13、圖 4-14 分別為三台、四台、五台、六台伺服器實驗時的一秒內的最大流量。

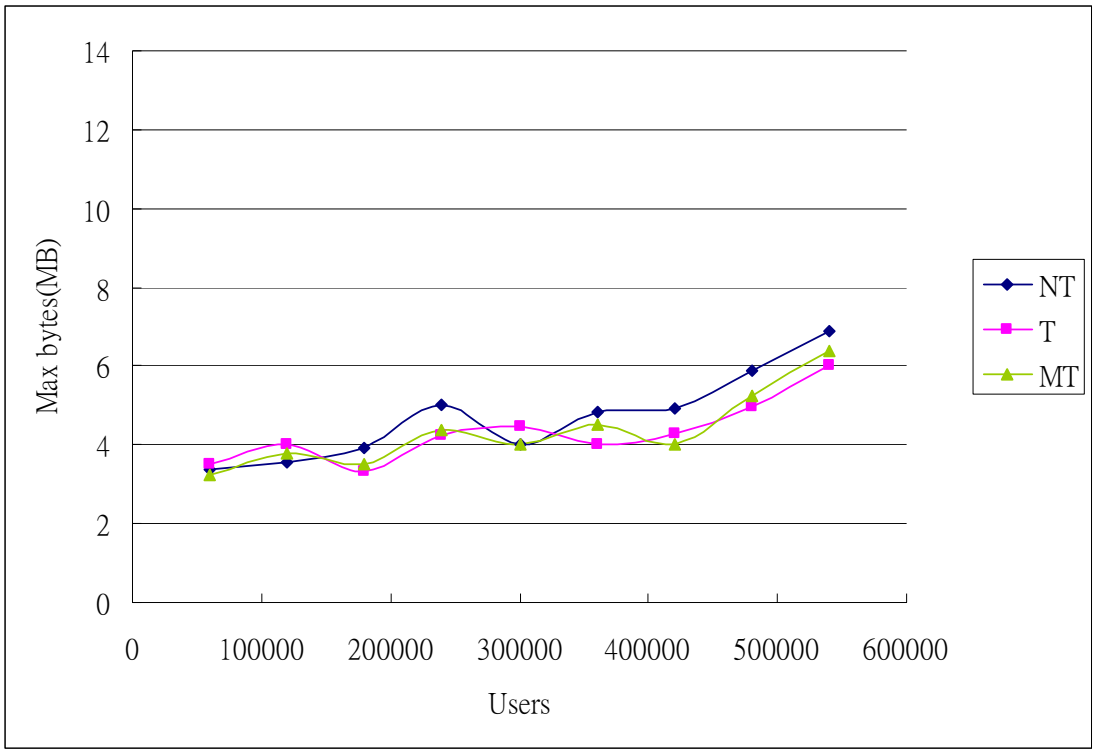

圖 4-11 三台伺服器時，一秒內最大流量與使用者數的分佈圖

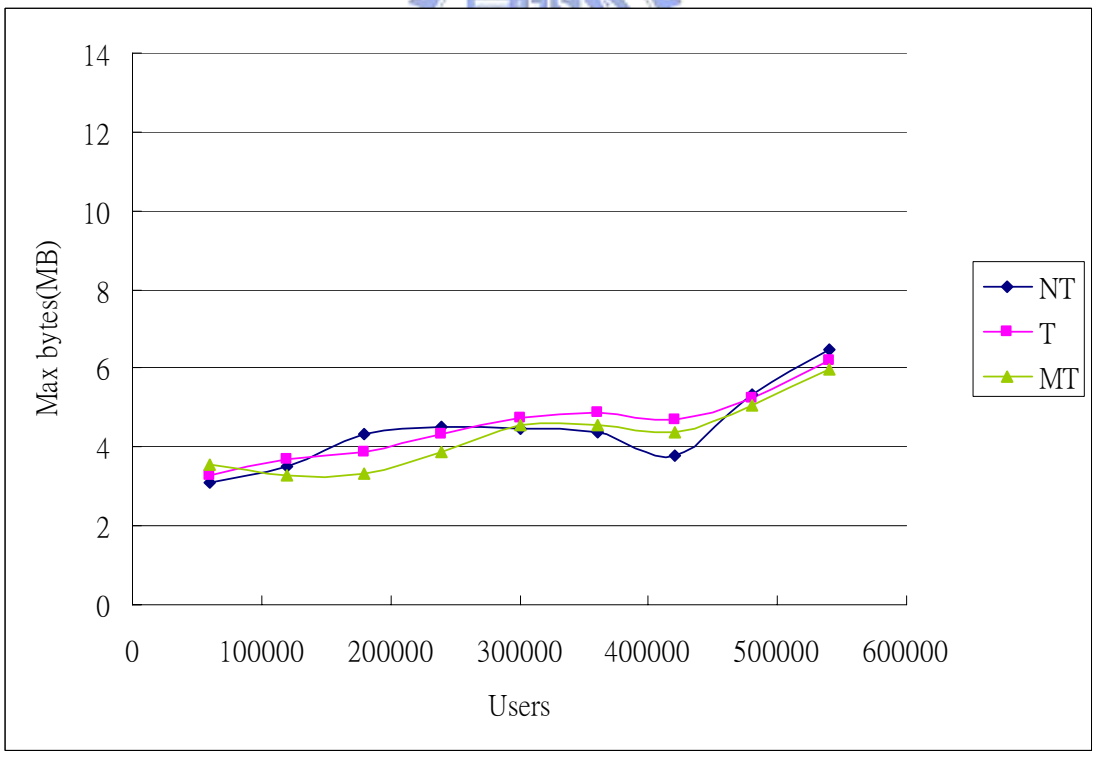

圖 4-12 四台伺服器時，一秒內最大流量與使用者數的分佈圖

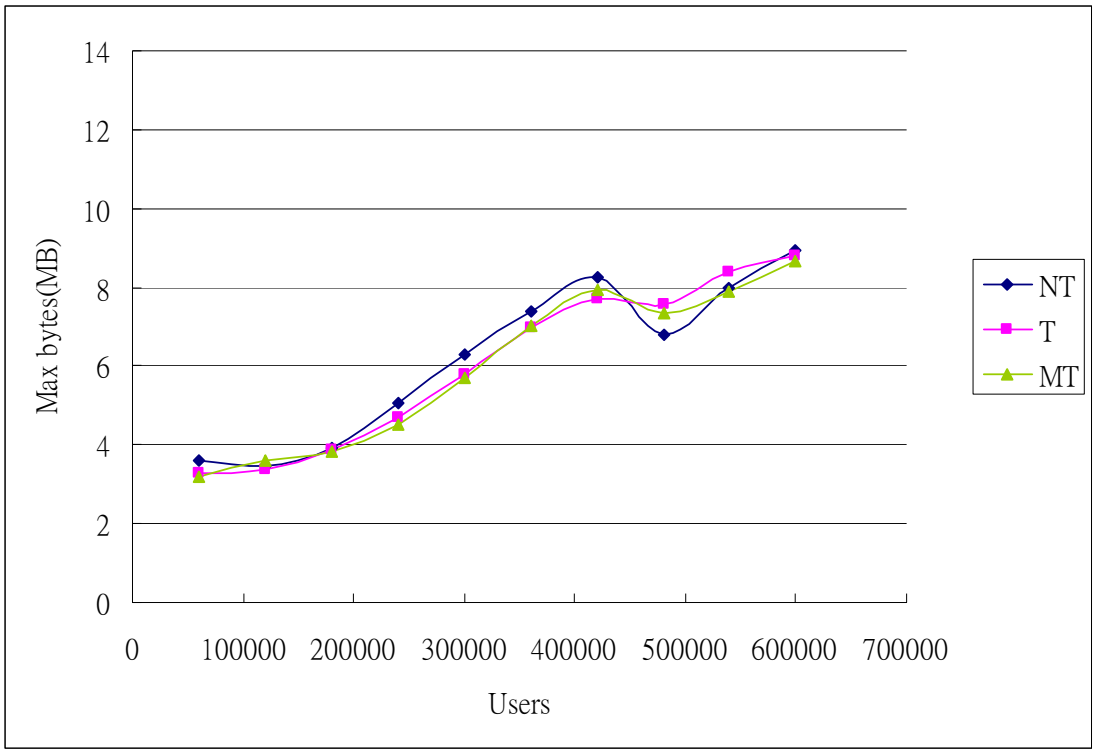

圖 4-13 五台伺服器時，一秒內最大流量與使用者數的分佈圖

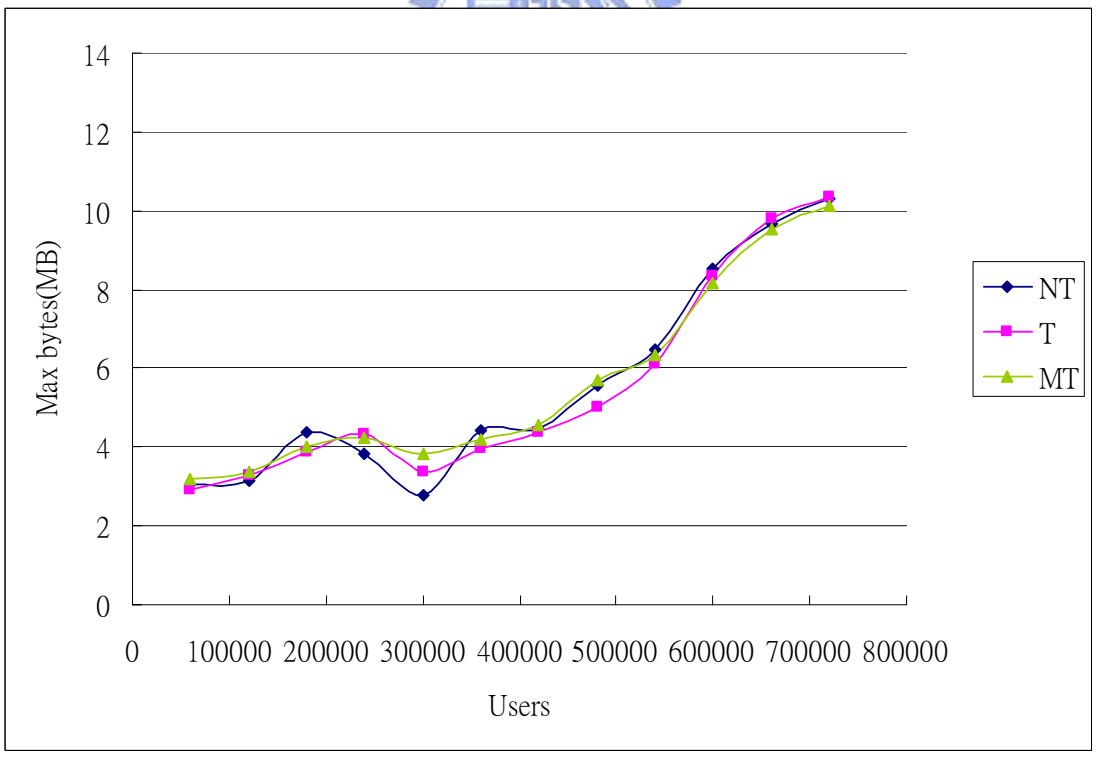

圖 4-14 六台伺服器時，一秒內最大流量與使用者數的分佈圖

我們可以看到在六台伺服器的情形中，網路的負載量已經快達到100Mbit，大致來說，三種情形的對外流量都很接近，且尚未超過極限，如此我們所得到的結果才有意義，而其中我們所改良的方法具有最好的效果。

第五章　結論與未來展望

由實驗證明，架設額外內部區域網路及分散型清單維護還有使用聊天專用伺服器都足以降低系統的效能，而 UDP 因為實作 ACK 的部分，由於等待 ACK 的關係造成延遲，大量的訊息傳輸使用 UDP 並不會比 TCP 的效果好。另外我們所提出的改良 Talk Server 方法，由實驗證明有較佳的結果，能支援的最大人數是最多的。前 10%高的平均回應時間在 3 秒內的情形下，一台伺服器時能支援 120000 人，增加到六台伺服器時可支援到 660000 人。

本論文對於測試的資料是採取網路封包量最龐大的時候來模擬，未來可考慮仔細分析線上即時傳呼系統的使用者訊息分佈，以機率統計的觀點來產生測試的資料，再進行測試，使用更多伺服器來進行實驗，測試結果便會更加接近實際情形。

參考文獻

[1] AOL Instant Messenger Home Page. http://www.aim.com/index.adp.

[2] J. Lemon. “Kqueue - a generic and scalable event notification facility.” In Proceedings of the USENIX Annual Technical Conference, FREENIX Track, June 2001.

[3] M. Day, Lotus, et al., “A Model for Presence and Instant Messaging”, RFC 2778, February 2000.

[4] M. Day, Lotus, et al., “Instant Messaging / Presence Protocol Requirements”, RFC 2779, February 2000.

[5] ICQ Inc., “ICQ Instant Messenger”, available from http://web.icq.com/, 2003.

[6] Microsoft Corporation, “MSDN Library”, available from http://msdn.microsoft.com/library/, 2003.

[7] Microsoft Corporation. “MSN Messenger.” http://messenger.msn.com.tw/.

[8] Ousterhout J. Why Thread Are A Bad Idea (for most purposes). Invited talk at the 1996 USENIX Technical Conference. http://home.pacbell.net/ouster/threads.pdf.

[9] R. Parviainen and P. Parnes. “Mobile instant messaging.” 10th International Conference on Telecommunications, Vol. 1, pp. 425 – 430. March 2003.

[10] D. Quan, K. Bakshi, and D. Karger. “A Unified Abstraction for Messaging on the Semantic Web.” Submission to The Twelve World Wide Web Conference 2003.

[11] D.C. Schmidt, Stal M, Rohnert H, Buschmann F. Pattern-Oriented Software Architecture Vol 2: Patterns for Concurrent and Networked Object. Wiley Computer Publishing, 2000.

[12] W.R. Stevens. UNIX Network Programming Vol 1: Networking API: Sockets and XTI (2nd edn). Prentice Hall PTR, 1998.

[13] W.R. Stevens. Advanced Programming in the UNIX Environment. Addison-Wesley, 1992.

[14] M.K. McKusick, K. Bostic, M.J. Karels, J.S. Quarterman. The Design and Implementation of the 4.4BSD Operation System. Addison-Wesley, 1996.

[15] D.C. Schmidt. “Reactor: An Object Behavioral Pattern for Concurrent Event Demultiplexing and Event Handler Dispatching.” Pattern

Languages of Program Design. Addison-Wesley, 1995.
[16] Yahoo! Inc., “Yahoo! Messenger”, available from http://messenger.yahoo.com/, 2003.
[17] Mike Mintz. “MSN Messenger Protocol“ , available from http://www.hypothetic.org/docs/msn/index.php
[18] H. Isaksoon, Version 5 of the ICQ Protocol, ICQ protocol specification document April 2001. Available from: http://www.algonet.se/~henisak/icq/icqv5.html - CTC.
[19] S. Leggio, T. Kulve, O. Riva, and M. Kojo. An Analysis of Instant Messaging and E-mail Access Protocol Behaviour in Wireless Environments. Technical report, University of Helsinki, Department of Computer Science, March 2004.
[20] Jeremy Hamman, AIM/Oscar Protocol. Available from: http://www.geocities.com/smokeyjoe12345/OscarProtocol.htm
[21] Mintz Mike. MSN Messaging Protocol description. Unofficial, second draft. Available from: http://mono.es.gnome.org/imsharp/tutoriales/msn/book1.html (last checked: 3.6.2003).
[22] R. Oettinger, Total Time Spent Using Instant Messaging Jumps 110 Percent At Work And 48 Percent At Home Versus Last Year, Reports Jupiter Media Metrix. Available from: http://www.jupiterresearch.com/xp/jmm/press/2001/pr_111401.html.