國立交通大學

多媒體工程研究所

碩 士 論 文

以 P2P 技術為基礎的巨量發佈系統

-客戶端與效能分析研究

The Study of the Massive Deployment System based on P2P Technology

- Clients and Performance Analysis

研 究 生：周鼎量

指導教授：吳毅成 教授

中 華 民 國 九 十 七 年 九 月

以 P2P 技術為基礎的巨量發佈系統

-客戶端與效能分析研究

The Study of the Massive Deployment System based on P2P Technology

- Clients and Performance Analysis

研 究 生：周鼎量　　Student：Ting-Liang Chou

指導教授：吳毅成　　Advisor：I-Chen Wu

國 立 交 通 大 學

多 媒 體 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Multimedia Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

September 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年九月

以 P2P 技術為基礎的巨量發佈系統
-客戶端與效能分析研究

研究生：周鼎量　　指導教授：吳毅成

國立交通大學 多媒體工程研究所

摘要

我們發展出一套以 P2P 技術為基礎的巨量發佈系統，用來解決一般遊戲公司在發佈遊戲程式或更新檔時，伺服端所需要的額外大量頻寬負擔問題。

此系統名稱為 Massive Deployment System (MDS)，其包含伺服器端與客戶端元件，而本論文探討客戶端的設計與此系統的效能分析。

The Study of the Massive Deployment System based on P2P Technology - Clients and Performance Analysis

Student: Ting-Liang Chou Advisor: I-Chen Wu

Institute of Multimedia Engieering
National Chiao Tung University

Abstract

We present a file deployment system based on P2P technology, which can greatly save the bandwidth of server when the company releases a game file or a patch file.

Our system named “Massive Deployment System” (MDS) which contains components of server side and client side. This thesis focuses on the design of client and performance analysis in ths paper.

誌謝

首先， 我要感謝我的指導教授，吳毅成教授，感謝老師讓我有機會參與這份研究計畫，並且總是很有耐心的對我指導，明確的提出錯誤部份，也讓我學到了不少的知識，也漸漸的明白了正確的研究方法與態度。

接下來，我要感謝實驗室的 P2P 小組的所有成員，因為有你們，這份計畫才能順利的完成，其中，我要特別感謝我的主要工作伙伴，哲毅，在合作的期間總是給予我相當多的幫助與指導，從你身上我真的學到了很多東西。 然後還有冠翬，在實驗的環境設計也幫了我非常大的忙，如果沒有你們的鼓勵與幫忙，我想我沒有辦法順利的完成這份研究。

感謝實驗室的學長，學弟，同學們，兩年的相處雖然短暫，但大家所帶來的歡樂，讓我在研究生之路上，解了不少苦悶。

感謝我的父母，在我求學生涯裡，總是給我許多精神上的幫助。

最後，感謝我的女友，貞妃，總是一直陪伴在我身邊，給予我關懷與鼓勵，才讓我能順利的堅持到最後的最後。

謹以此論文，獻給每位支持我的人。

目錄

表目錄

圖目錄

第一章、緒論

本章將介紹本論文的動機背景及論文的研究目標，並且會在最後簡介本論文的內容大綱。在 1.1 節裡，我們會說明研究動機與目的，1.2 節提出研究目標。最後，1.3 則對本論文做個簡要的章節介紹。

1.1 研究動機與目的

隨著網路的發達，玩網路遊戲的玩家也愈來愈多，然而當一個網路遊戲公司想發佈新的遊戲檔案或更新檔時，其瞬間的上傳需求量可能很大。對於大型的遊戲公司而言，瞬間的下載者人數可能到達數萬人以上。在這種情況下，若使用傳統 Client-Server 架構的遊戲公司，想要提供不錯的下載速度，則需要租用非常大的頻寬來提供服務，以避免玩家在第一時間內，因為下載不夠快而選擇玩其它遊戲。

然而公司向 ISP 業者來租用大型頻寬是非常昂貴的，月租費用多則可達到數十萬到百萬不等，相較之下，一般玩家玩遊戲時，對頻寬的需求反而不高。

這裡簡單的舉出一個例子，假設現在有一千個玩家，他們在玩遊戲時需要的頻寬量約為 30Kbps 左右，然而當玩家下載一個遊戲程式時，會期望能有較快速的下載速度，如今假設需求量約為 256Kbps 左右，如此一來，兩者的總頻寬需求各為 30Kbps*1000 = 30Mbps 及 256Kbps*1000 = 256Mbps。接下來再以現今每 Mbps 約三千塊台幣的價錢來初步估計，則兩者所需要花費的價錢各為每個月七十七萬與九萬元台幣。

由上面一個簡單的例子可得知，公司為了解決瞬間下載量所付出的成本，是相當驚人的，而且可以想像的是，這些額外租用的頻寬，

很可能在一般時期用不到一半，如此對公司而言也是相當的浪費。

因此，為了解決傳統 Client-Server 架構下， 伺服端負擔過重的情形，我們希望建構出一個以 Peer-to-Peer 為基礎的檔案傳輸系統 (Massive Deployment System，簡稱 MDS)，讓客戶端在下載的過程及遊戲進行過程中，能夠提供部份的上傳，以此減輕傳統伺服端負擔過重的情況。

最後，我們希望可以模擬出線上遊戲公司發佈檔案時的下載環境，比較 MDS 系統與傳統 Client-Server 架構下，玩家們的下載速度，並且分析可以讓公司在減少多少程度的頻寬下，仍保有與原本近似的服務品質。

1.2 研究目標

為了建立以 Peer-to-Peer 為基礎的檔案發佈系統，我們需要實作出一個能有效分享檔案片段機制的客戶端程式，如此才能讓客戶端下載檔案更快速。而我的目標則是實作此客戶端程式，並模擬不同頻寬的玩家在下載檔案時，採用 MDS 系統與傳統 Client-Server 架構下，玩家們的下載速度比較，並分析公司在採用 MDS 系統時，能節省多少的頻寬，來提供與原本近似的服務品質。

1.3 論文章節介紹

首先，我們會在第二章簡介幾種 P2P 軟體的發展，並對我們主要參考的 BitTorrent 及我們所提出來的 MDS 系統架構作介紹，接下來在第三章將介紹 MDS Client 端的實作部份。之後在第四章則會模擬遊戲公司在發佈檔案時的網路環境，比較採用不同架構的系統時，不同網路環境的玩家們的下載速度。最後在第五章對本論文作個總結，並提出一些未來的發展工作。

第二章、背景說明

本章節將介紹幾種常見的 Peer-to-Peer 的檔案分享系統及 Massive Deployment System (MDS) 系統架構介紹。首先我會在 2.1 節簡介幾個著名的 P2P 軟體，並針對我們所參考的 BitTorrent 作比較仔細的檔案分享流程。之後在 2.2 節再介紹 MDS 系統的架構及元件說明。

2.1 P2P 檔案分享系統

Peer-to-Peer 簡稱 P2P，為現今網路運用的熱門話題之一，其主要的概念為，擺脫傳統 Client-Server 的架構，利用客戶端們的上傳頻寬來減少伺服器端的負擔，並使得檔案分享變的更有效率。

隨著網路的進步，資訊與資料的傳播速度也跟著加快，尤其是近年來 P2P 技術的發展，讓大家更能輕易的分享的檔案，相較於過去的網路能力，如今的下載速度能讓使用者在分享檔案時將更不受限於檔案的大小，接下來將介紹幾個著名的 P2P 檔案分享系統，首先是 2.1.1 節會對 Napster [1]作簡介，接下來於 2.1.2 節簡介 eDonkey，最後在 2.1.3 節則會針對 BitTorrent 作較詳細的介紹。

2.1.1 Napster

年僅十九歲的 Shwan Fanning 在 1999 年創造了第一個應用 P2P 技術的音樂分享服務，讓許多人能透過網路交換 MP3 音樂檔案，然而也因此受到了唱片公司的控告，於 2000 被法院下令停止服務，但卻也開啟了之後眾多 P2P 檔案分享服務出現。之後 Napster 在 2003 年改以提供付費的音樂下載服務營運。

2.1.2 eDonkey2000

eDonkey2000 [2]簡稱 eDonkey 或 ed2k，在 2000 時由 Jed McCaleb 所創造，其使用 MD4 摘要演算法來讓每個檔案擁有唯一性，並讓客戶端透過一個中央伺服器來尋找檔案，之後再讓客戶端彼此連線來進行檔案的傳輸。特別的是，eDonkey 首創將檔案切割成片段的機制，讓客戶端能夠在下載完部份片段後，即時的分享目前所擁有的片段，如此一來，大大的提升了客戶端頻寬的使用，使得分享的速度更為有效率。然而，eDonkey 在 2005 年也面臨了 RIAA (Recording Industry Association of America) 的控告問題。

2.1.2 BitTorrent

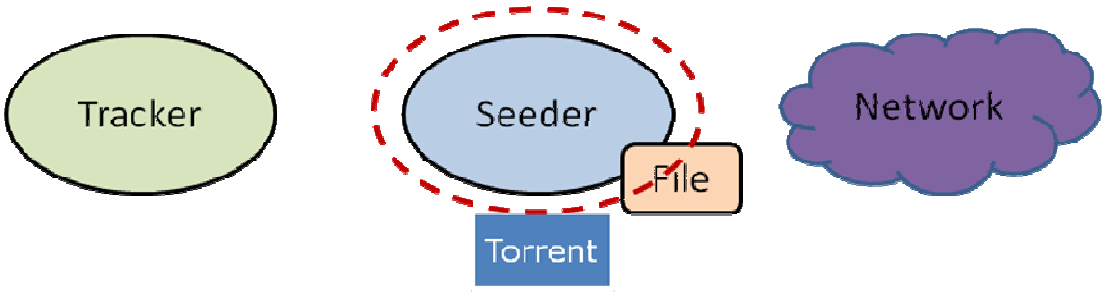
BitTorrent (BT) [3]為現今最流行的 P2P 軟體之一，其特性為針對網際網路環境的特色，來設計出更有效率的傳輸策略，加速檔案分享的過程 [4][5][6][7]，其主要元件定義如下：

- 檔案－客戶端所要分享的檔案資料，可以是單一檔案或是多層資料結構。
- 片段－原始檔案在 BT 的分享流程中，將會被視為多個片段的交換，當客戶端擁有所有的片段才算是拿到完整的檔案。
- 子片段－在實際網路上，檔案是以更小的子片段單位在作交換，當客戶端擁有某片段的所有子片段時，才算是拿到完整的片段。
- Tracker－透過 http 連線來提供客戶端互相連線的名單，使客戶端能找到目前正在下載特定檔案的群組並加入。
- Torrent File－針對檔案所產生的一個簽章，主要包含了下載檔案的相關資訊，如針對整體檔案的 hash 值，檔案大小，片段大小，片段 hash 值等，另外還有 Tracker 的 URL，使客戶端可以透過 Tracker 來進行連線。
- Leecher－尚未擁有所有片段的客戶端

- Seeder－擁有完整檔案的客戶端。

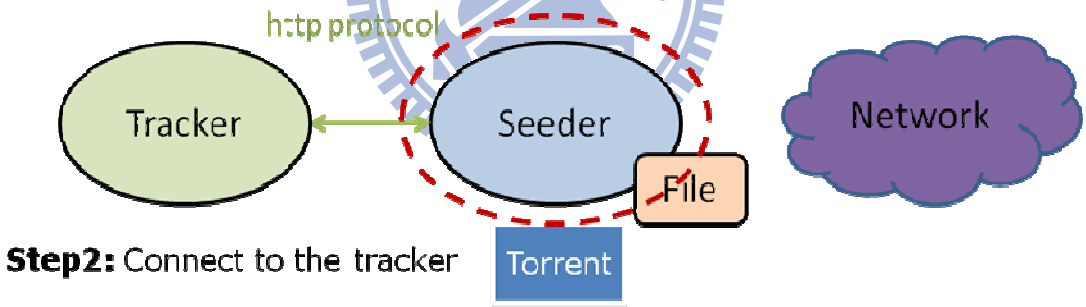
假設當某一個原始檔案者想要發佈檔案時，其流程如下：

圖 1：BT 檔案發佈者依據檔案製作 Torrent File



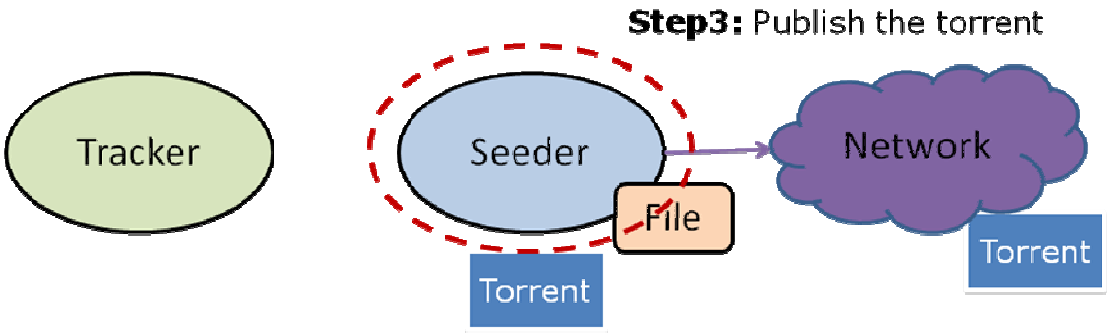
Step1: Make a torrent

圖 2：BT 檔案發佈者向 Tracker 註冊 Torrent 之 hash 值



Step2: Connect to the tracker

圖 3：網路散佈 Torrent File 給其它客戶端



假設當某一個客戶端想要下載此檔案時，其流程如下：

圖 4：BT 客戶端從網路取得 Torrent File

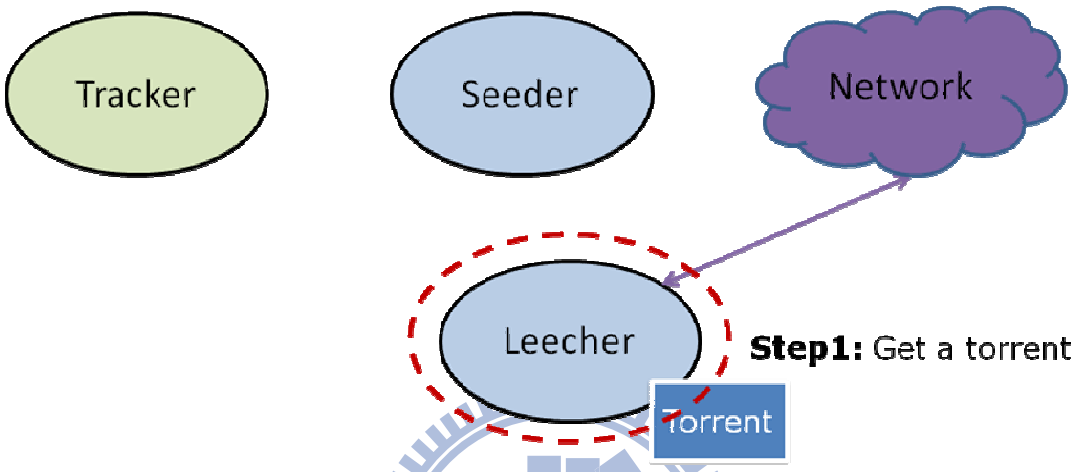


圖 5：BT 客戶端利用 Torrent File 向 Tracker 取得連線名單

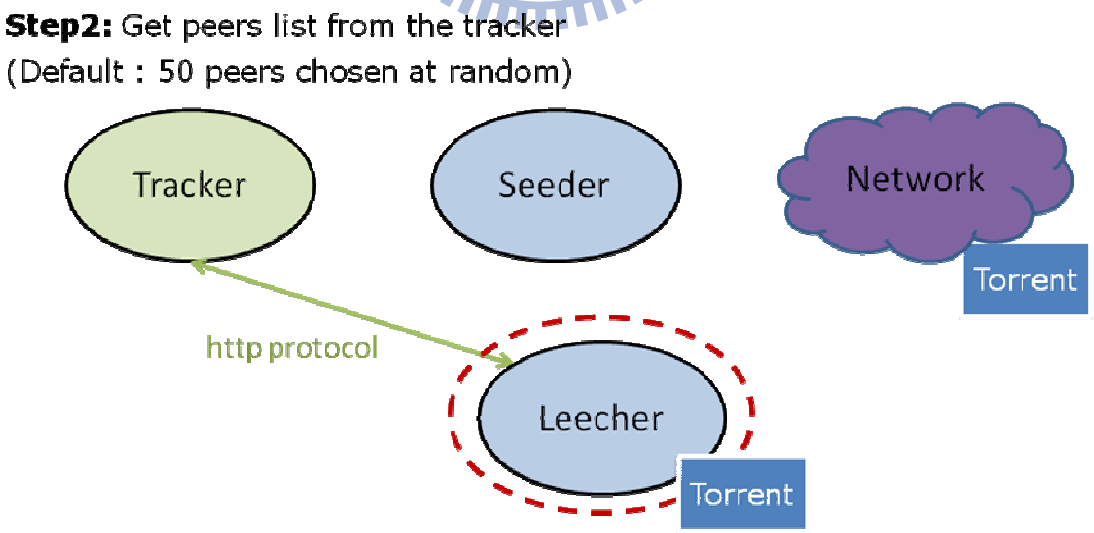
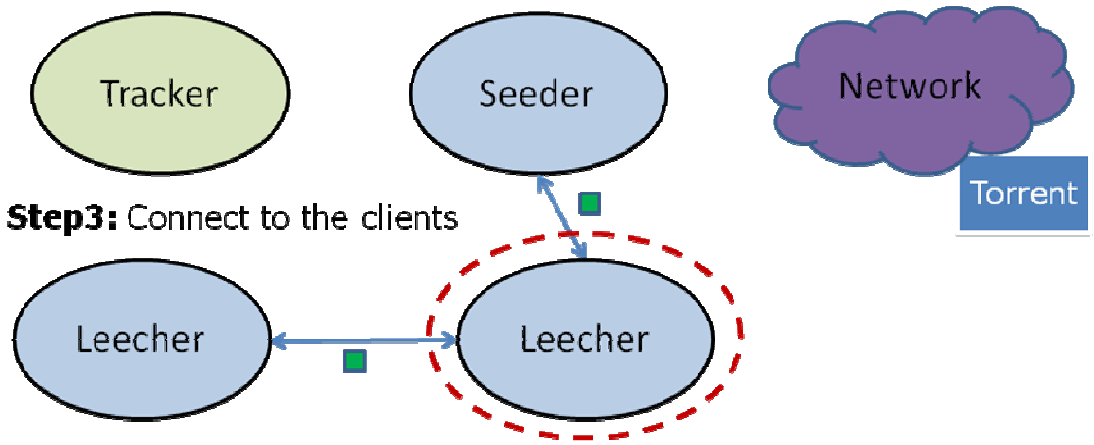


圖 6：BT 客戶端與其它客戶端建立連線並開始下載作業



2.2 Massive Deployment System (MDS)

下圖為我們所設計出來的 MDS 系統架構圖，其中右上方的虛線部份為，公司內部的操作元件與 MDS 核心元件的切割。

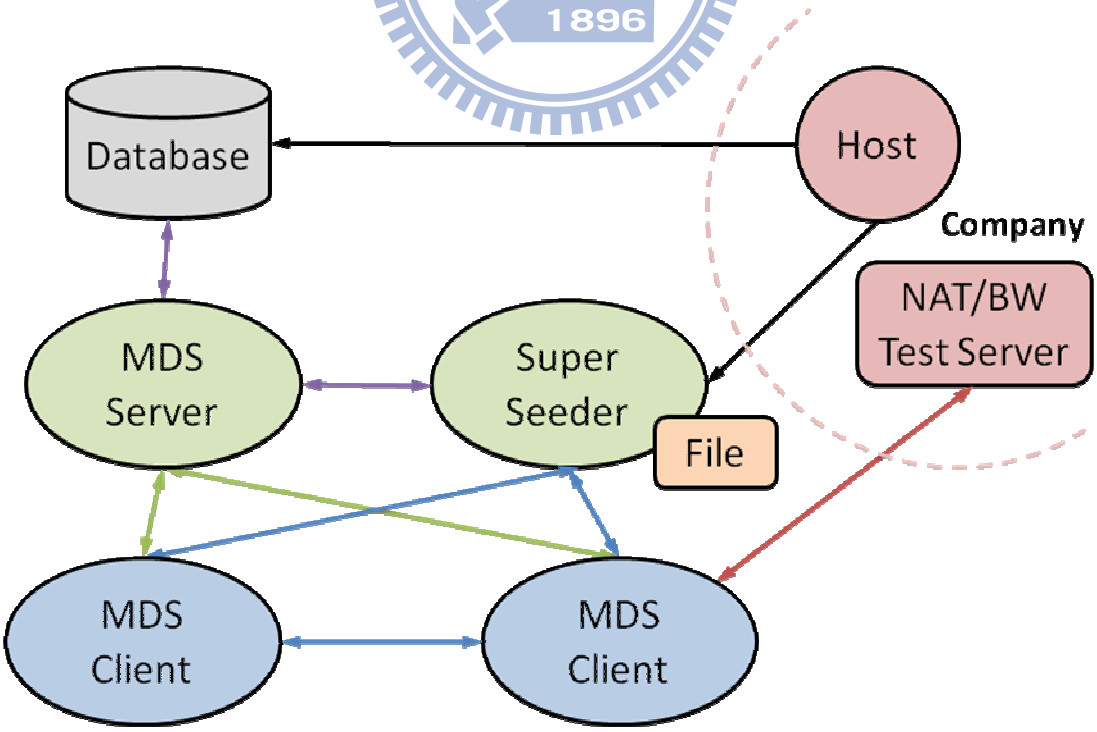


圖 7：MDS 系統架構

接下來將介紹 MDS 系統元件，圖中元件功能定義如下：

- File－遊戲公司所想要發佈的遊戲檔案或更新程式。
- File Signature－針對特定檔案的簽章，類似 BT 裡面的 Torrent file 角色功能。
- Host－遊戲公司內部的操控，決定檔案發佈的開始，其主要功能為製作 File Signature 後，再將其發佈到 Database，並將原始檔案傳送到 Super Seeder。
- MDS Server－主要功能為發送 File Signature 給 Super Seeder 及 MDS Client，監控所有連線對象，並提供 MDS Client 連線名單。
- Super Seeder－由公司所提供的原始檔案上傳者，如果客戶端之間沒有互傳行為的話，則可將其視為傳統 Client-Server 架構中的伺服端。當 Super Seeder 拿到完整檔案後，會先登入到 MDS Server，接受 MDS Server 的控管並等待 MDS Client 的連線。
- MDS Client－擁有檔案片段交換能力的客戶端程式，其採用的通訊協定為 BitTorrent 協定。
- Database－儲存 Fire Signature，MDS Server，MDS Client，Super Seeder 名單以及相關的 log 或回報資訊。
- NAT/BW Test Server－提供 MDS Client 作 NAT 及頻寬測試伺服器。

關於 MDS 系統發佈檔案的詳細流程及系統架構，則可參考另一篇相同主題的論文“*The Study of Massive Deployment System Based on P2P Technology – The Servers and System Architecture*” [8]。

第三章、MDS Client 實作

此章節將介紹我們如何實作 MDS Client，首先，在 3.1 節，我們會簡介我們實作 MDS Client 時所採用的 BT Client，然後於 3.2 節介紹 BT 協定。最後在 3.3 節談到我們如何將 BT Client，與 MDS 系統作結合。

3.1 CTorrent

在實作 MDS Client 上，我們希望能讓 MDS Client 快速交換檔案片段，以縮短下載作業時間，因此我們採用了檔案交換效率較高的 BT 協定，來實作檔案片段交換機制的部份，其中我們選用 CTorrent [9] [10] (一個 BT Client 的實作程式) 為修改對象，而 CTorrent 的主要特點如下：

- Open Source。
- 撰寫語言－使用 C/C++。
- 平台－Linux, FreeBSD, MacOS。
- 程式特色－輕量級程式，快速。

選用 CTorrent 的主要原因，除了它是用 C/C++所撰寫的 Open Source 之外，其為輕量級的程式，會比較易於開發與結合在我們所設計的 MDS 系統上。

3.2 BitTorrent 協定

本章節將介紹 BitTorrent 協定的主要訊息及通訊策略，首先會在 3.2.1 節介紹 BT 協定的通訊訊息，接下來在 3.2.2 節介紹 BT 協定的通訊策略中的上傳策略，最後在 3.2.3 節介紹下載策略。

3.2.1 BitTorrent 協定通訊訊息

BitTorrent [11]協定規範了 BT Client 之間的通訊協定，其中客戶端在第一次連線時會有一個 handshake 的動作，用來再次確認是否雙方採用同樣的協定，以及是否正在針對同一個 torrent file 作下載，之後再藉由規範的通訊訊息，來交換彼此的片段資訊，狀態資訊等，以便於之後的片段交換，而客戶端之間主要的通訊訊息如下：

- Bitfield：一開始客戶端建立連線，完成 handshake 之後會先傳送一個 bitset，用來告知對方我目前擁有哪些片段。
- Have：當客戶端收集到一個完整的片段時，會先判斷是否跟 Torrent File 裡的片段 hash 值相同，若資料確定無誤時，則再對每一個連線對象發送此訊息，告知大家要更新我所擁有的片段資訊，如此一來，每個客戶端就可以很清楚的知道，目前的連線對象擁有哪些片段。
- Interest：當客戶端發現對方擁有自己缺少的片段時，發送此訊息表示想從對方下載片段，若此狀態不改變，則不會再重複送出此訊息。
- Un-interest：當客戶端發現對方無自己缺少的片段時，發送此訊息表示不需從對方下載片段，唯有當客戶端對連線對象的狀態從 interest 變成 Un-interest 時才發送，若此狀態不改變，則不會再重複送出此訊息。
- Choke：當客戶端端從所有對自己 interest 的對象中，挑選出上傳對象後 (根據上傳策略，將於 3.2.2 節介紹)，將會對剩下的連線對象發送此訊息，表示此時並不提供上傳給對方，若此狀態不改變，則不會再重複送出此訊息。
- Un-choke：當客戶端端從所有對自己 interest 的對象中，挑選出上傳對象後 (根據上傳策略，將於 3.2.2 節介紹)，發送此訊息，表示此時將提供上傳給對方，唯有當客戶端對連線對象的狀態從 choke 變成 Un-choke 時才發送，若此狀態不改變，則不會再重複送出此訊息。

- Request：當客戶端發現對方 un-choke 自己時，將會挑選出想要下載的子片段 (根據下載策略，將於 3.2.3 節介紹)，並發送此訊息告知對方目前自己所需要的子片段。
- Piece：當客戶端端收到對方所傳送的 request 訊息時，會將子片段以此訊息送出 (注意，此處雖然是 Piece，但實際上是子片段)。

3.2.2 BitTorrent 協定上傳策略

BT 原作者在訂出最原始的 BT 協定，與上下載策略後，花了將近一年的時間在作參數最佳化的調整，以得到較好的傳輸效能。其中 BT 協定預設每個客戶端同時的上傳對象有四人，而名額的分類如下：

- Regular Un-choke：佔客戶端上傳的三個名額，並於每 10 秒後作一次輪替。
- Optimistic Un-choke：佔客戶端上傳的一個名額，並於每 30 秒後作一次輪替。

其中 Regular Un-choke 的選法，會根據目前是 Seeder 或 Leecher 時期，而有不同的選擇方法如下：

- Leecher 時期：秉持著一報還一報 (Tit for tat) 的精神，當對方對我的上傳貢獻度愈高，我將愈優先貢獻上傳給對方。而客戶端主要為依據過去 20 秒內的平均上傳速度及總貢獻度來挑選出前 3 名當 Regular Un-choke。
- Seeder 時期：並不需要它人的貢獻，其會優先上傳給下載速度快的連線對象。

值得一提的是，原始 Seeder 時期的作法，可能會導致一個惡意不提供上傳的 Leecher，因為下載速度夠快，而佔據掉 Seeder 的上傳，最後導致整體系統的傳輸效能下降，針對此點，CTorrent 在 Seeder 的時期上傳則是平均的分佈給所有的連線對象。

至於 Optimistic Un-choke 的選擇，則是從沒被 Regular Un-choke 選到的連線對象中，用隨機的方式來選取出一位，其主要用意為以下兩種：

- 提供上傳給完全沒有片段的客戶端：當客戶端剛開始下載檔案時，因為身上沒有任何的片段，無法與其它的 Leecher 作交換，因此可藉由這個機會來取得第一個片段。
- 尋找有潛力的連線對象：客戶端在 Leecher 時期可藉由 Optimistic Un-choke 來主動提供上傳給新的連線對象，如此則有機會找到貢獻度較高的連線對象。

3.2.3 BitTorrent 協定下載策略

BT 下載策略主要分為以下幾個原則：

- Strictly Priority：當客戶端在選擇下載的子片段時，會優先選擇盡快完成某一個片段，如此才更有本錢跟其它連線對象交換。
- Rarest First：當客戶端在選擇下載的片段時，可以利用目前的連線資訊，來優先選擇比較稀有的片段，如此一來，整體下載速度才不會在後半段時，因為某些片段的稀少，而導致下載速度受限於那些片段的傳輸，使得整體分享效能變慢。
- Random First Piece：當客戶端身上沒有任何的片段時，將會用隨機的方式來挑選第一片要下載的片段，以避免一開始就直接挑選來源可能很少的片段，而導致取得第一片的下載速度更慢。
- Endgame Mode：當客戶端在下載的過程中，發生了某幾個片段的下載速度變慢的情形時，因為同時仍會有其它的片段持續在下載，因此對於客戶端的整體下載速度不會有太大的影響。然而如果是發生在下載最後的片斷時，則很可能因為這樣導致最後下載的時間拉長，因此當客戶端在下載最後一個片段時，可以向不同的連線對象要求相同的子片段，而當客戶端收到其中一個人所傳

送的子片段時，則會通知其它重複發送 Request 的連線對象，不需要再傳送這一個子片段過來 (Cancel)，以避免頻寬的浪費。

3.3 MDS Client 實作上的修改

本章節將介紹如何對 CTorrent 作修改，使其能與 MDS 系統整合。首先，會在 3.3.1 節介紹設計與 MDS Server 溝通模組的部份，接下來於 3.3.2 節介紹如何設計 MDS Client 的控制協定，使其能與遊戲公司的程式溝通，以便於結合。

3.3.1 MDS Server 溝通模組

此章節將介紹與 MDS Server 溝通的模組。我們將在 3.3.1.1 節介紹 MDS Client 在開始與其它 MDS Client 端連線之前的初始化流程，在 3.3.1.2 節裡討論發送 MDS Client 名單要求的時機，接下來在 3.3.1.3 節介紹重新登入的機制，最後在 3.3.1.4 介紹回報機制。

3.3.1.1 MDS Client 初始化流程

MDS Client 在開始連線前的流程如下：

步驟一：MDS Client 開始啟動並連線到 NAT 及頻寬測試伺服器，以得到自己 NAT 型態，以及上傳頻寬能力的資訊。

步驟二：MDS Client 從原本的 MDS Server 連線名單裡面，用隨機 (避免掉部份 MDS Server 負載平衡的問題) 的方式挑選出一個 MDS Server 準備登入。

步驟三：MDS Client 告知 MDS Server 目前自己手上的 File Signature 版本。

步驟四：MDS Server 透過登入資訊，來判斷 MDS Client 是否拿到目前線上服務的 File Signature，若否，則傳送一份給 MDS Client。

步驟五：MDS Client 利用最新的 File Signature 來檢查自己所擁有的檔案片段，並將 MDS Client 詳細的資訊 (如 NAT，頻寬，監聽連線的通訊埠，是否為 Seeder 等資訊) 傳送給 MDS Server。 (此步驟與原本 BT 客戶端登入 Tracker 的步驟類似)

步驟六：MDS Server 傳送給 MDS Client 有關 Super Seeder，其它 MDS Client 與 MDS Server 的連線名單。(此步驟與原本 BT 客戶端收到 Tracker 資訊的步驟類似)

步驟七：MDS Client 開始與 Super Seeder 及其它 MDS Client 進行連線。 (此步驟與原本 BT 客戶端的啟動相互連線步驟類似)

3.3.1.2 MDS Client 連線名單要求

一般傳統的 BT 客戶端，會主動向 Tracker 要求連線名單的時機，有以下三種情況：

- 第一次與 Tracker 連線：如之前 BT 下載流程所提到的，當客戶端重新啟動時會先向 Tracker 要求連線名單。
- 固定時間逾時：當客戶端發現與 Tracker 隔一段固定的時間內沒通訊時，會主動與 Tracker 要連線名單 (如，CTorrent 預設時間為 30 分鐘)。
- 連線狀況不佳：當客戶端發現連線數量過低時，會向 Tracker 要求一份新的連線名單。

然而，對於 MDS 系統而言，如果 MDS Client 沒有連線到 Super Seeder，則很有可能會有很大的影響，因為 Super Seeder 很可能為

客戶端主要的檔案片段來源之一。因此，我們希望在 MDS Client 發現與 Super Seeder 斷線時，能立即向 MDS Server 要到新的 Super Seeder 名單以建立連線。

3.3.1.3 MDS Client 狀態回報

為了讓 MDS 系統的中央控制伺服端 MDS Server 可以監控 MDS Client 們的下載情況，以及達到整體的效能分析，MDS Client 提供兩個主動的回報機制如下：

- Completed report：當 MDS Client 下載完成時，將會通知 MDS Server 下載的開始與結束時間，以及上下載的總量等資訊。
- Regular report：定期回報給 MDS Server 最近二十秒內的上下載速度及連線情況等資訊。

3.3.2 MDS Client 控制協定

為了讓 MDS 系統可以與一般公司的系統作結合，我們設計了一個 MDS Client 控制協定，此協定透過本地端 TCP 連線來進行通訊，經由此介面，外部其它程式可以來控制 MDS Client 的部份行為，其概念如以下示意圖：

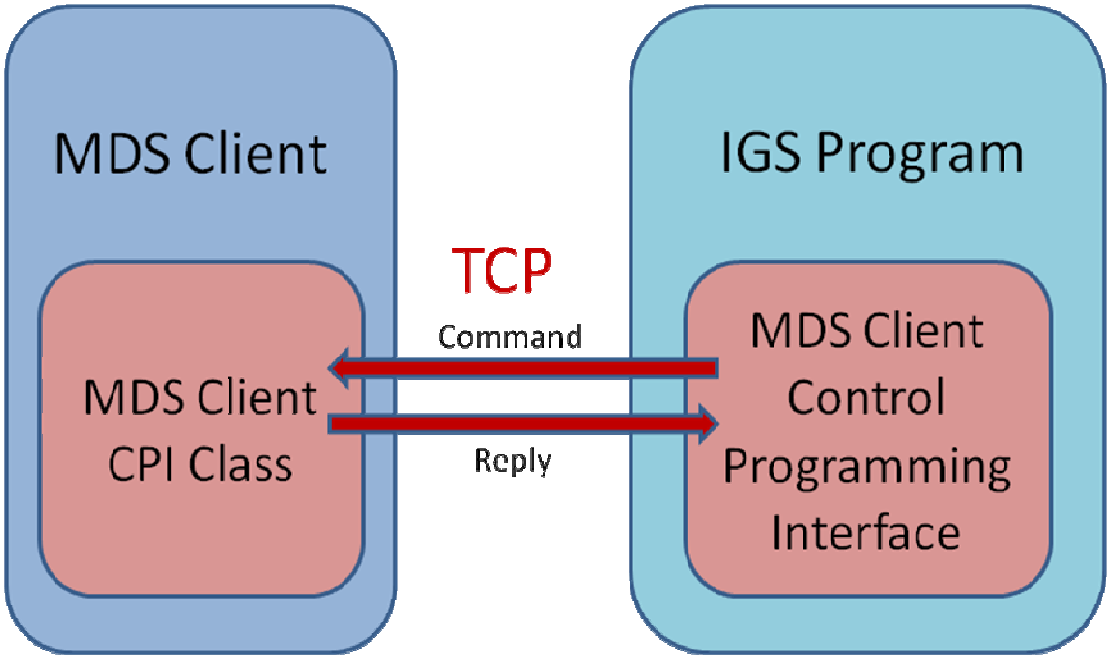


圖 8：MDS Client 控制協定

上圖中兩個物件皆為存在客戶端的程式，其中左邊藍色方型為 MDS Client，裡面提供了 Control Programming Interface (CPI) 的物件來讓右邊的遊戲公司的執行程式 (如，鈊象公司的遊戲大廳程式)，來進行操控，其中控制介面如下：

- 下載行為控制：下載暫停，下載繼續，結束程式。
- 下載速率控制：最大上傳速率，最大下載速率。
- 資料回報：下載進度回報，上下載速度回報，上下載總量回報，連線情況回報。

第四章、實驗結果

本章節將會模擬玩家們在下載檔案時的網路環境，分析在不同系統下的下載速度，以及針對 MDS 系統在採用不同的上傳策略時，對玩家們下載速度所產生的影響。在 4.1 節，將會先介紹實驗環境與模擬玩家頻寬及公司頻寬的設定，接下來在 4.2 節，我們將比較採用傳統 Client-Server 架構與 MDS 系統架構，在固定的伺服端上傳頻寬下，所各別得到的玩家整體下載速度，並分析 MDS 系統在使用多低的頻寬下，仍可得到與原本近似的服務品質。在 4.2 節談到當改變 MDS Client 上傳策略時對整體下載速度的影響。最後，在 4.3 節談到當改變 Super Seeder 上傳策略時對整體下載速度的影響。

4.1 模擬環境設定

為了模擬遊戲公司進行發佈檔案的過程，我們首先假設公司在發佈一個更新檔時，是以總上傳頻寬 40Mbps 來服務 1000 個玩家的瞬間下載，基於上面的假設，我們將作以下的實驗設定：

- 更新檔大小：18MB。
- 客戶端人數：100 人。
- 伺服端上傳頻寬：4Mbps (對應玩家人數比例縮小 10 倍)。
- 連線順序：同時連線 (為達到實驗公平性，當所有客戶端都連線到 MDS Server 時，MDS Server 才回傳連線名單，讓客戶端開始下載程序)。
- 連線數量：在傳統 Client-Server 架構下，所有的客戶端只會與 Super Seeder 連線，而在 MDS 系統架構，則讓客戶端多連其它九個客戶端 (因為在真實環境下，不可能所有的客戶端皆互相連線，因此我們縮小客戶端的連線總量)。

關於客戶端的 NAT [12]設定，則是參考群想網路科技公司的內部

資料(Public IP 與在 NAT 後面的數量約為一比一) [13]，而頻寬資訊則是基於目前家用 ADSL 頻寬來作假設，而詳細的客戶端設定如下圖：

UL Rate	DL Rate	Public IP (%)	NAT (%)
0KBps	48KBps	2	2
8KBps	48KBps	8	8
16KBps	96KBps	8	8
32KBps	192KBps	8	8
64KBps	384KBps	8	8
128KBps	768KBps	8	8
256KBps	1536KBps	8	8

表 1: 客戶端實驗環境設定

上圖的客戶端頻寬分佈，是參考目前家用 ADSL 不同的上傳頻寬能力，所設定出的七個區間。而區間分佈則是以倍數成長的方式，來作區分。首先假設上傳能力最差的玩家們，其上傳頻寬介於 0Kbps ~ 8Kbps，第二差的玩家群則介於為 8Kbps ~ 16Kbps，第三區間的玩家們為介於 16Kbps ~ 32Kbps，以此類推，以倍增的方式來設定出七個區間。

接下來取各區間的下限值當作玩家上傳能力的設定，例如上傳能

力介於 0Kbps 到 8Kbps 的玩家，將會被當作沒有任何上傳能力，而第二差的玩家們上傳頻寬則假設為 8Kbps，以此類推。

而各區間的玩家下載頻寬設定，則是直接以上傳頻寬乘 6 倍的方式來得到 (參考家用 ADSL 實用上的比例)，例如上傳能力最強的玩家，其下載能力將設定為 256*6=1536Kbps，以此類推。然而此處的例外為上傳頻寬為 0 的玩家，其下載頻寬與第二差的玩家們設定值相同。

至於各區間的玩家人數設定，則是先不考慮頻寬最弱區間的玩家(此區間的玩家相對的少)，以平均分配各區間人數比例的方式來得到，之後再將分配剩下的人數設定為最弱區間的玩家，例如上傳頻寬為 16Kbps，下載頻寬為 96Kbps 的玩家，在 Public IP 及 NAT 各佔 8%。

最後，為了實驗上的加速，我們將所有的上傳速度改以 KByte per second (KBps) 為基本單位來進行實驗，其中為了等比例讓客戶端的交換率升高，我們將原本 Regular unchoke 的週期從 10 秒變成 2 秒，Optimistic unchoke 的週期從 30 秒變成 6 秒。

4.2 Client-Server 與 MDS 系統架構比較

根據 4.2 節模擬玩家網路的環境設定，我們將比較採用傳統 Client-Server 與 MDS 系統架構下的下載速度比較。

首先，我們先介紹一個假設伺服端上傳無限的特殊情況，在這種情況下，客戶端的下載時間將會被本身的下載速度所限制，例如下載頻寬最差的玩家們，其最快的下載時間為 18MB/48KBps=384sec，以此類推，畫出所有客戶端下載速度最快的理論值，並以完成時間由短到長作排序，如下圖所示 (我們將此線稱為 Unlimited)：

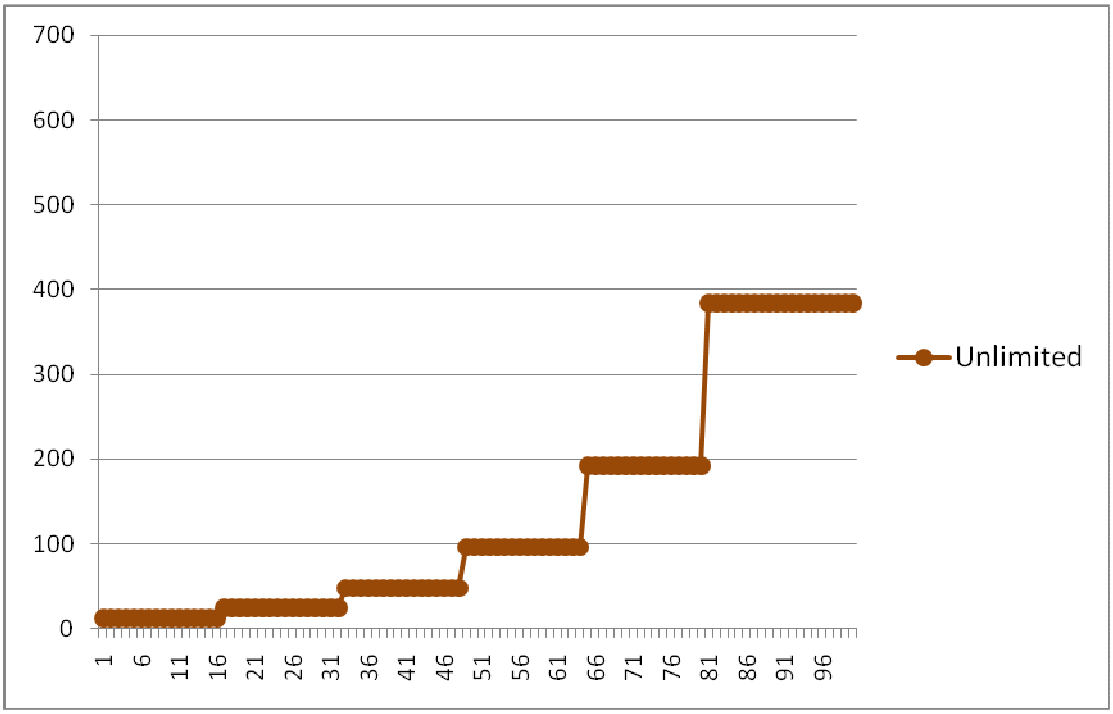


圖 9：客戶端下載速度最快理論值

上圖的縱軸代表下載時間，橫軸代表客戶端總數，由上圖可以看到由於每個區間的下載頻寬皆為倍數差距 (除了最後的兩群客戶端下載頻寬皆為 48KBps 之外)，因此下載時間也成倍數成長。

接下來我們利用計算分配頻寬的方式，來畫出傳統 Client-Server 架構下的下載時間 [14]。其算法是將 Super Seeder 的上傳頻寬，依據各客戶端的下載頻寬，來等比例分配。

首先，先假設最低頻寬的客戶端群，其下載速度為 X，第二低下載頻寬的客戶端群，其下載速度為 2X，以此類推，則最高下載頻寬的客戶端群，其下載速度為 32X，接下來再把每個區間的客戶端人數乘以其下載速度，來得到各區間的總下載量。例如下載速度最低的人數共為 4 + 16 = 20 人，而其下載速度皆為 X，因此可得到下載總量為 20X，以此類推可得到下表：

DL Rate	客戶端數量	單一下載速度	區間整體下載量
1536K	16	32X	512X
768K	16	16X	256X
384K	16	8X	128X
192K	16	4X	64X
96K	16	2X	32X
48K	20	X	20X

表 2：Client-Server 架構下載速度理論值

接著我們可以計算出所有客戶端的總下載速度為 512X+256X+…+16X+4X = 1012X，之後再根據 Super Seeder 上傳頻寬為 4000KBps 來得出 X=4000 KBps /1012 = 3.95KBps，求出 X 後便可得到實際下載速度。由於檔案大小為 18MB，所以可得出最快下載速度的客戶端，下載完成所花費的時間為 18MB/32*3.95KBps = 145.728sec。

之後其它客戶端的下載時間，則是先將尚未下載完成的客戶端們，扣掉已下載檔案的部份 (在第一批玩家下載完成的這段期間內，不同玩家所下載的檔案大小，依據其下載速度乘上時間即可得到)，接著，再如同之前的方式，讓 Super Seeder 的上傳頻寬以等比例的方式分配給尚未完成下載檔案的玩家，以此類推 (然而這部份需要注意到，Super Seeder 分配給各客戶端的頻寬，不能超過客戶端原本的下載速度)，可以得到以下的理論曲線 (我們稱之為 CS_th_4000，代表傳統 Client-Server 架構下，Super Seeder 上傳為 4000KBps 所得到的理論值)：

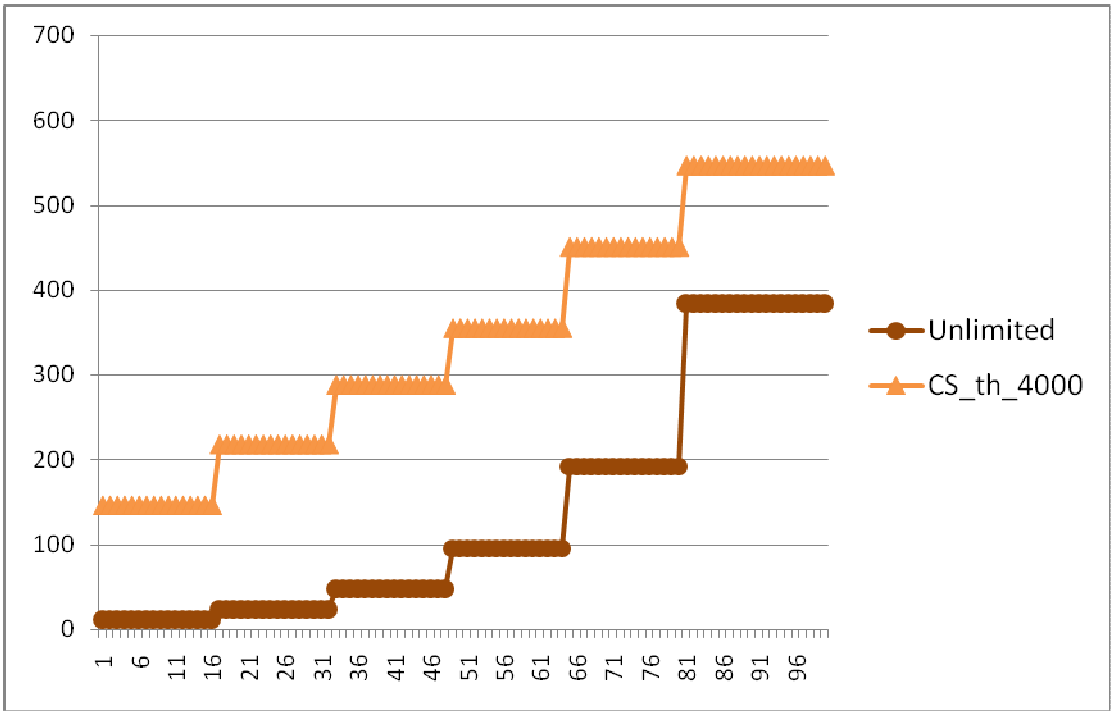


圖 10 : CS_th_4000

接下來，我們將介紹在 P2P 架構下，Super Seeder 上傳頻寬為 4000KBps 的最快理論，如下圖：

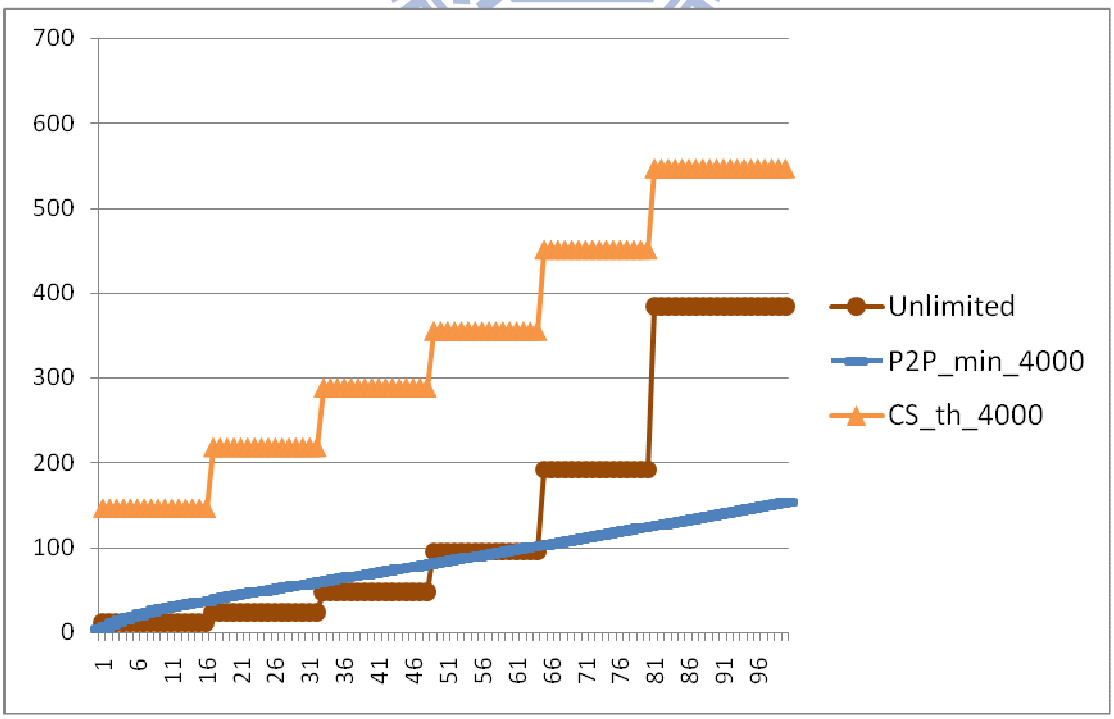


圖 11 : P2P_min_4000

上圖的理論值為參考其它論文的推導公式，套用實驗的環境設定，所計算出來的結果。其概念為不考慮客戶端下載頻寬的限制下，讓大家優先上傳給某個特定上傳頻寬大的客戶端，如此能讓整體的下載速度最佳化 (我們稱之為 P2P_min_4000，代表在 P2P 架構下，Super Seeder 若提供上傳頻寬 4000KBps，而 100 個客戶端上傳頻寬能力如前面所提到的設定，則可達到的最快完成時間)。接下來我們將實驗結果畫上(我們稱之為 P2P_4000，代表在 P2P 架構下，Super Seeder 提供上傳頻寬為 4000KBps 的實驗結果)，如下圖：

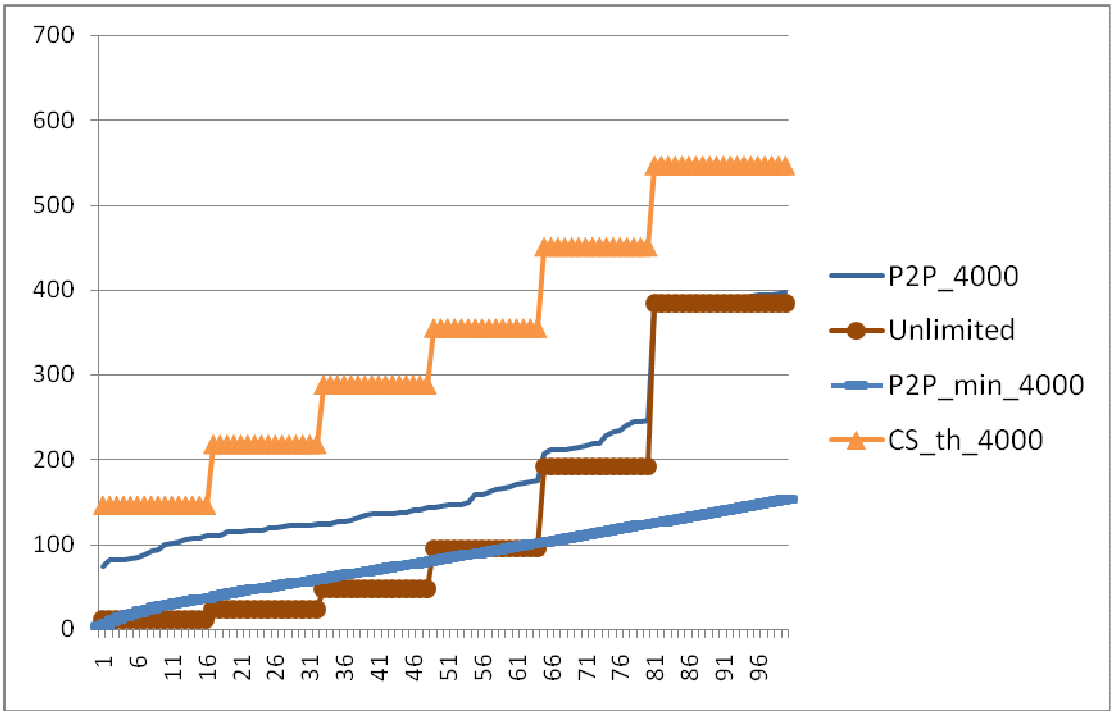


圖 12 : P2P_4000

由上圖可以看出，當 Super Seeder 為 4000KBps 時，使用 MDS 系統所得到的平均下載速度相對提高許多。接下來我們將以倍減的方式來減少 Super Seeder 的上傳頻寬，並用上圖呈現的方法 (同樣 Super Seeder 的頻寬設定下，P2P 理論值與實驗值為同色線條)，來展示結果。

當 Super Seeder 上傳頻寬為 2000KBps 時，實驗結果與 P2P 理論值，如下圖所新增的兩條紅線：

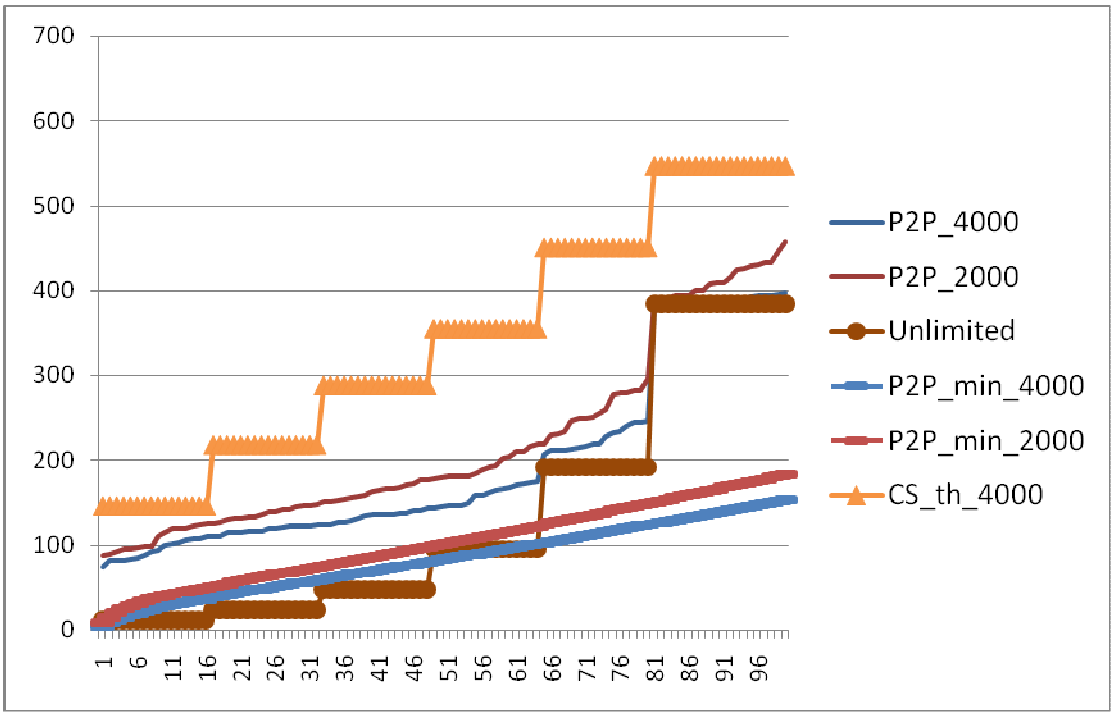


圖 13：P2P_2000 & P2P_min_2000

當 Super Seeder 上傳頻寬為 1000KBps 時，實驗結果與 P2P 理論值，如下圖所新增的兩條綠線：

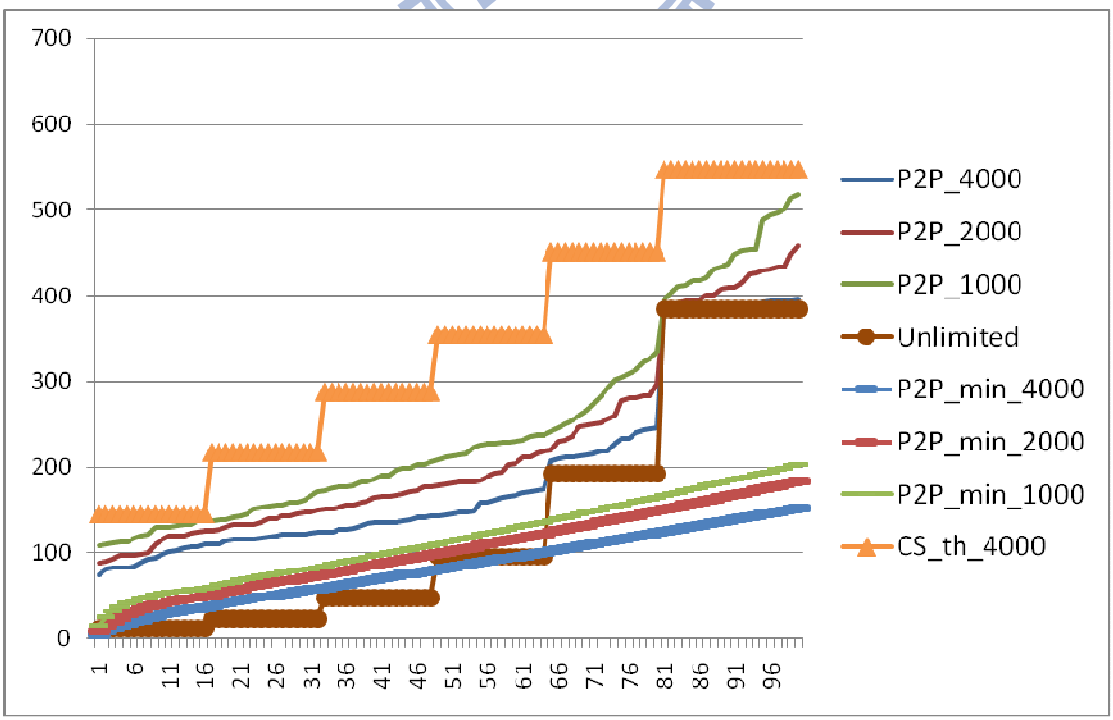


圖 14：P2P_1000 & P2P_min_1000

當 Super Seeder 上傳頻寬為 500KBps 時，實驗結果與 P2P 理論值，如下圖所新增的兩條紫線：

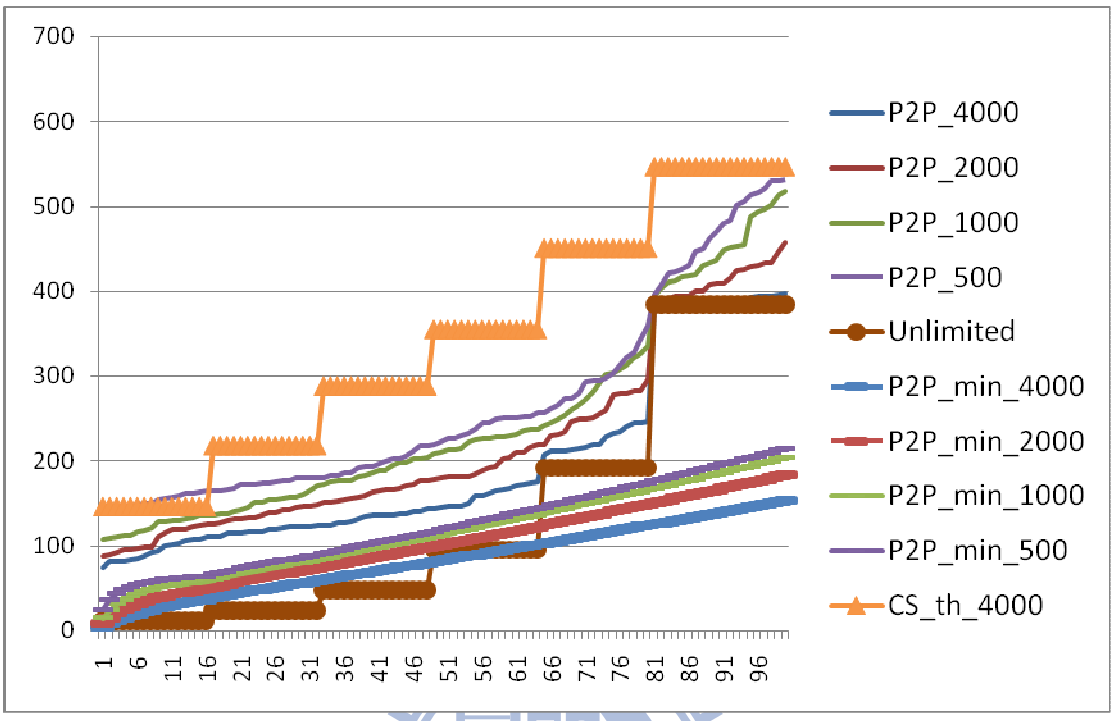


圖 15：P2P_500 & P2P_min_500

當 Super Seeder 上傳頻寬為 250KBps 時，實驗結果與 P2P 理論值，如下圖所新增的兩條青藍線：

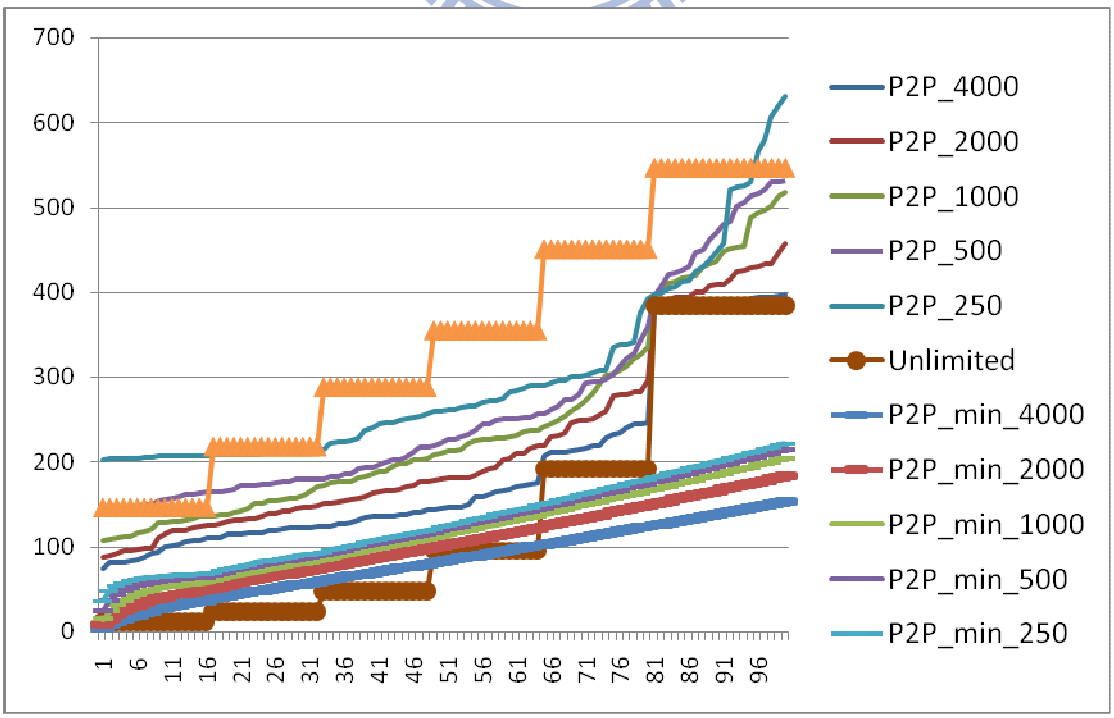


圖 16：P2P_250 & P2P_min_250

由上圖可以很明顯的看出，在採用 MDS 系統架構下，即使 Super Seeder 的上傳頻寬降到原本的 1/16，即 250KBps，仍能與原本傳統 Client-Server 架構的效能相近。

接下來為了要以客戶端角度，來觀察下載時間變化，我們將上述的實驗結果，以另一種圖表方式來呈現。首先我們將 CS_th_4000 當成基準線 (圖中 y 值為 1 的橫線)，再讓 MDS 系統在不同的 Super Seeder 上傳頻寬下，所得到的實驗結果除以基準線，來得到下載速度的比值，如下圖：

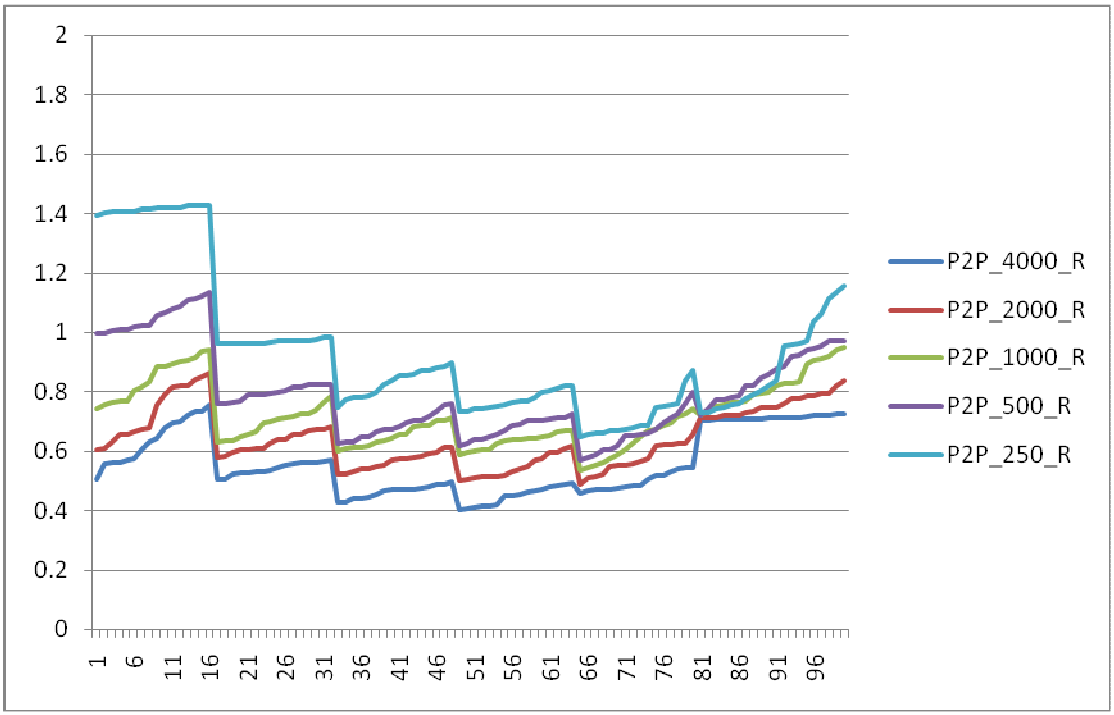


圖 17 ：P2P 與 Client-Server 架構下載時間比值

上圖中 P2P_4000_R 代表將 P2P4000 / CS_th_4000 所得到的結果，以此類推。由上圖可以很明確的看出，不同頻寬的客戶端，對於下載速度變化的感受。當使用 MDS 系統時且 Super Seeder 上傳頻寬為原來的 1/8，即 500KBps 時，絕大部份客戶端的下載速度都比原本來的快 (可從上圖中，紫色線條絕大部份都低於 1 來看出)。而當 Super Seeder 上傳頻寬為原本的 1/16，即 250KBps 時，由於伺服端的上傳頻寬太小，導至於下載頻寬最強的客戶端下載速度變

慢，平均下載時間約為原本的 1.4 倍 (可從上圖中，青藍色線條最前端部份，平均約為 1.4 來看出)，但其它的客戶端的平均下載速度仍比原本來的快。

4.2 MDS Client 上傳策略改變

此章節將比較，改變 MDS Client 的上傳策略後的下載速率比。根據之前所提到的 P2P 理論值計算概念，我們希望藉由優先上傳給，擁有較大上傳頻寬的客戶端，來加速整體下載時間。

因此在此章節，我們將對客戶端的上傳策略作改變，原本 Regular Un-choke 的選擇，其改變如下：

- Leecher 時期：將原本保持 Tit for Tat 的精神改成優先選擇上傳頻寬較大的客戶端，但為了避免造成部份客戶端 starving 的現象，以及加速最後片段的下載，若客戶端要求的片段為一開始的部份片段，或最後的部份片段，則有較高的優先權。
- Seeder 時期：將原本公平服務的原則改成優先選擇上傳頻寬較大的客戶端，但為了避免造成 starving 的現象，以及加速最後片段的下載，若客戶端要求的片段為一開始的部份片段，或最後的部份片段，則有較高的優先權。

而原本 Optimistic Un-choke 的選擇，則是從隨機挑選改成，讓上傳頻寬較大的客戶端，擁有較高的機率被選到。

接下來將對不同上傳頻寬的 Super Seeder 來作分析，此節的圖表分析將會類似 4.1 節的比較方式，來觀察改變客戶端上傳策略後的下載速度。

當 Super Seeder 上傳頻寬為 4000KBps 時，發現兩者的下載速度並沒有太大的差異，如下圖：

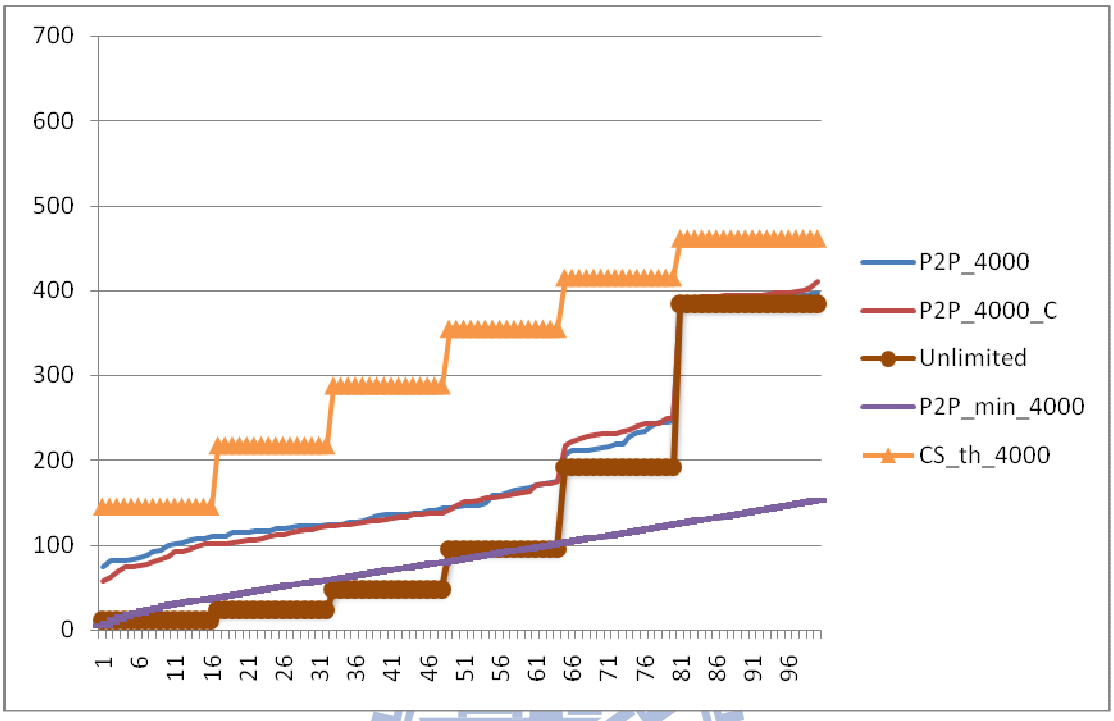


圖 18：P2P_4000_C

上圖中 P2P_4000 代表原本的下載速度，而 P2P_4000_C 則代表改變客戶端上傳策略時的下載速度，至於 Unlimited，CS_th_4000 與 P2P_min_4000 則為 4.1 節所提到的理論值，用來比較實驗結果，接下來類似的圖表也會將以相同的方法來表達。

接下來我們把上面的圖表改用比值的方式來呈現，如下圖：

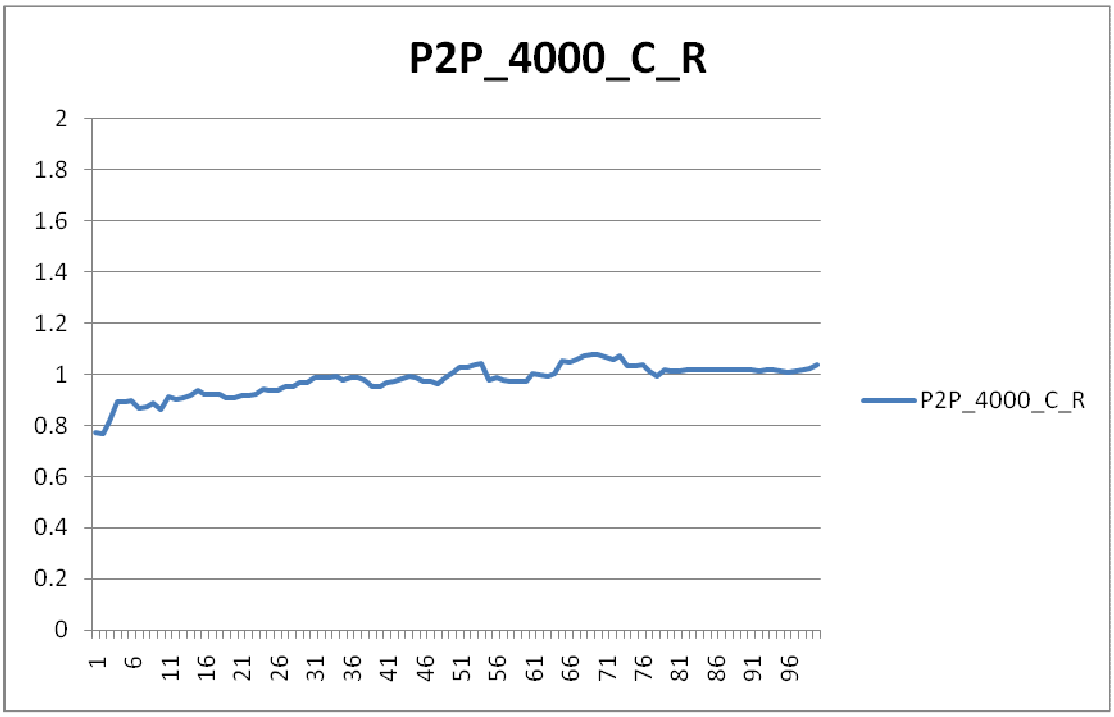


圖 19：P2P_4000_C_R

上圖中 P2P_4000_C_R 代表當改變客戶端上傳策略時，其下載速度除以原本的 P2P_4000 下載速度，所得到的比值，由上圖可看出兩者並沒有太大的差異。

當 Super Seeder 頻寬為 2000KBps 時，實驗結果如下圖：

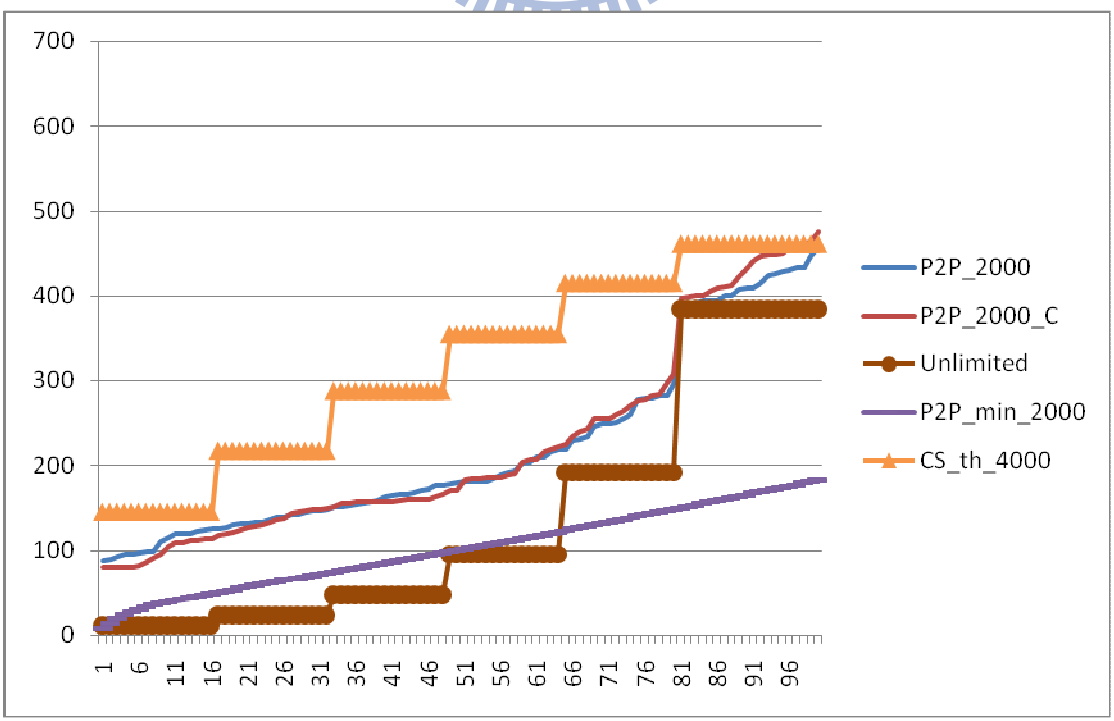


圖 20：P2P_2000_C

由上圖可以看出兩者的差異仍然不大，而比值分佈如下圖：

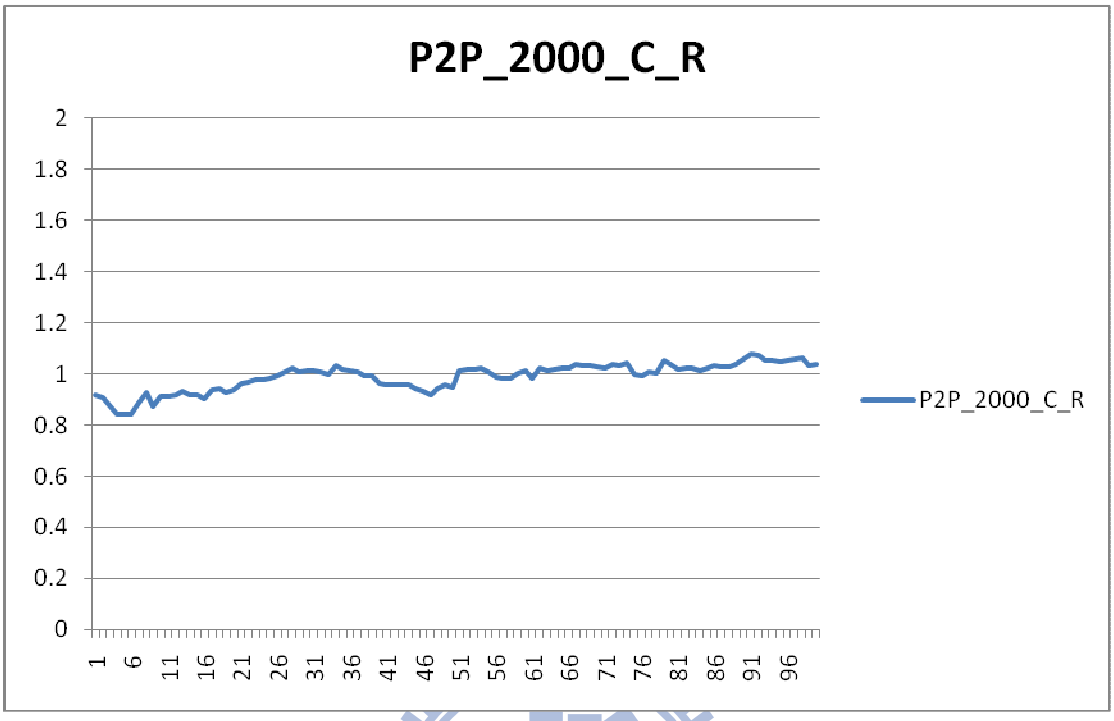
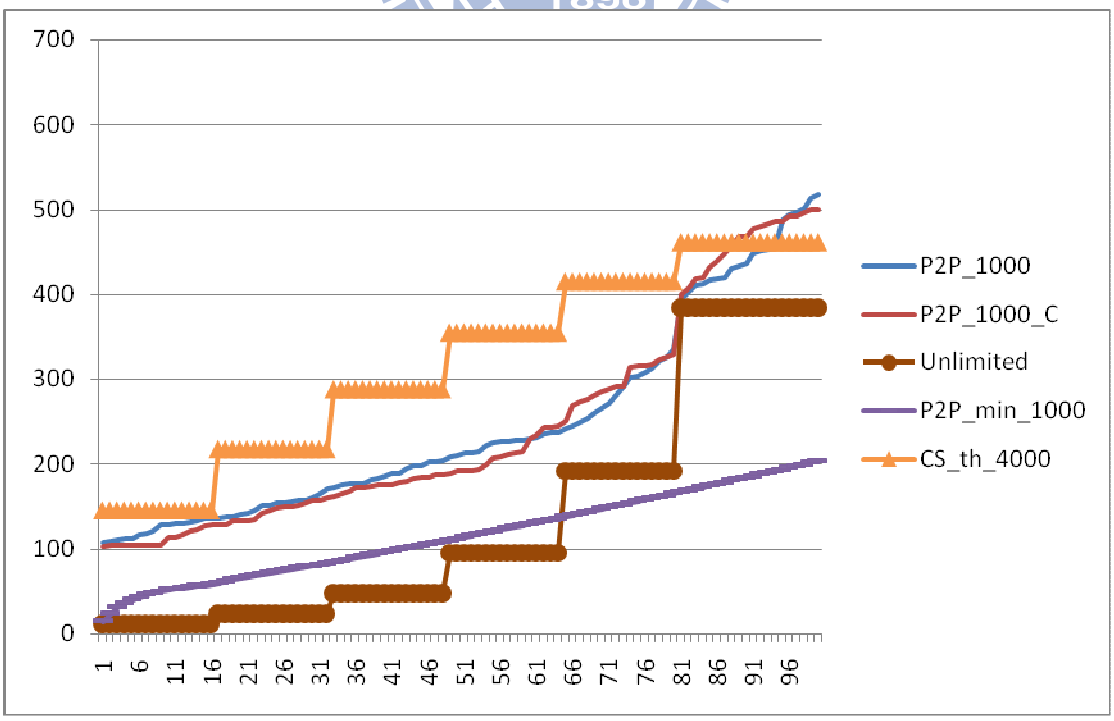


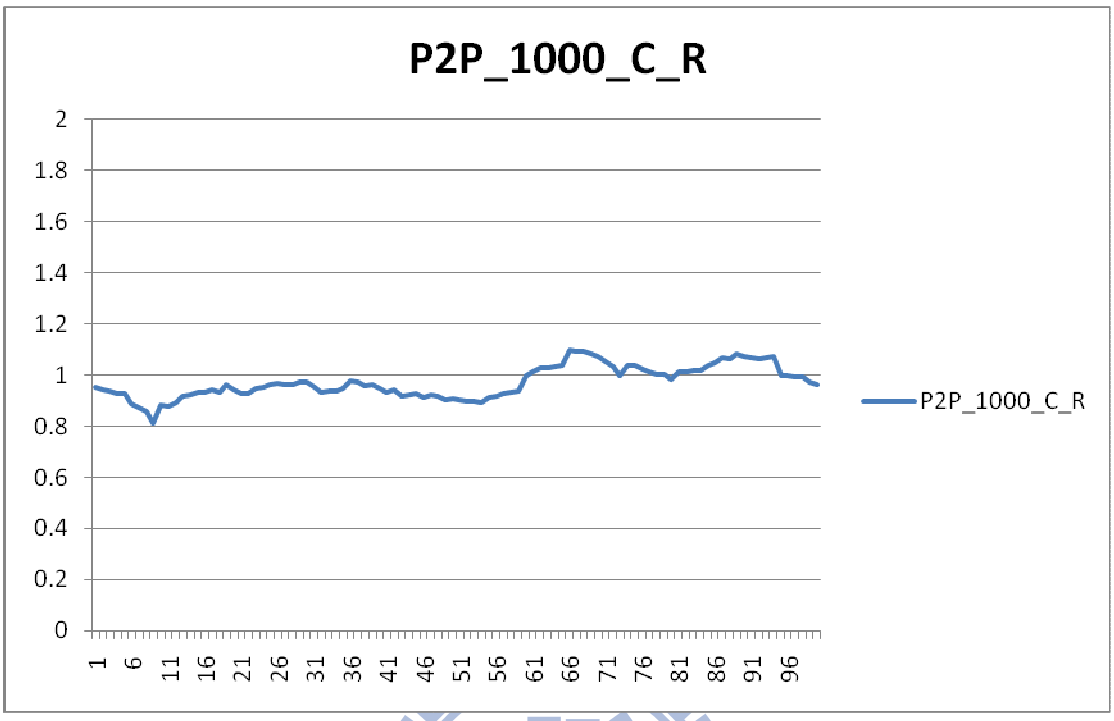
圖 21：P2P_2000_C_R

當 Super Seeder 頻寬為 1000KBps 時，實驗結果如下圖：



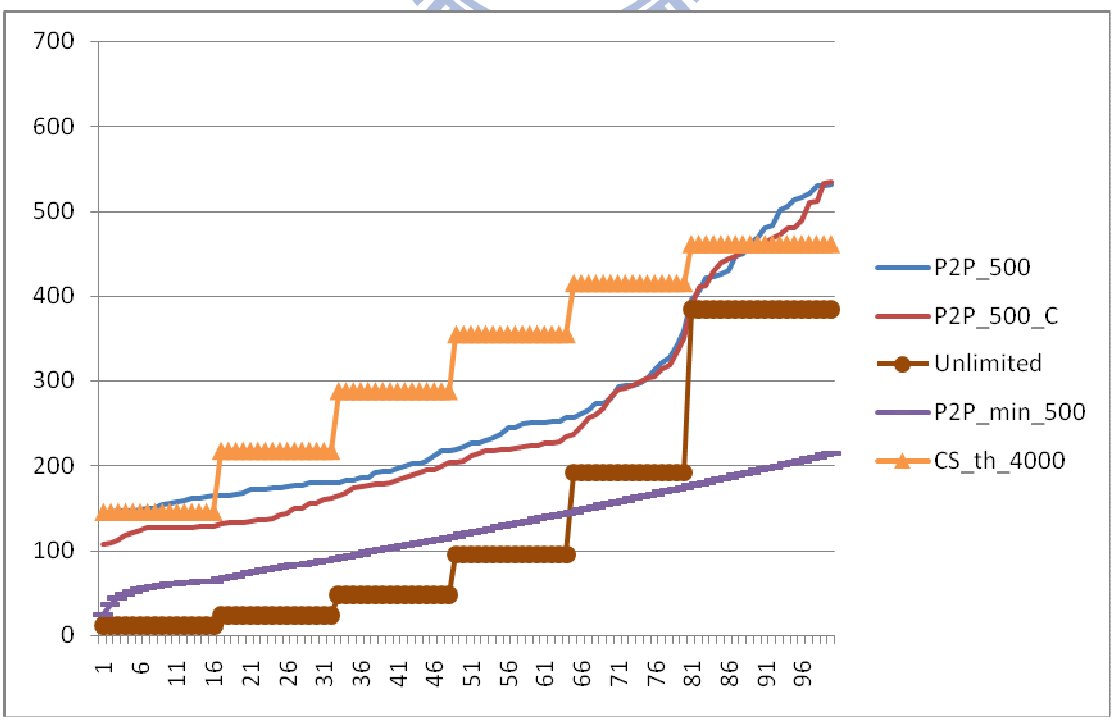
圖表 22 ： P2P_1000_C

由上圖可以看出兩者的差異依然不大，而比值分佈如下圖：



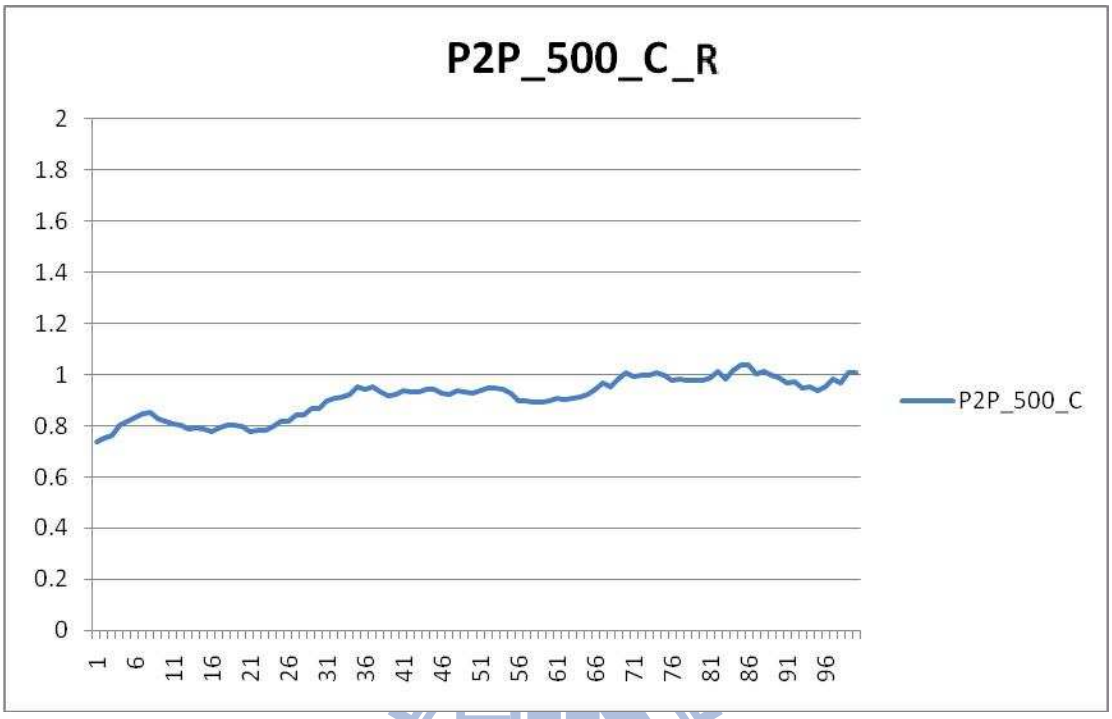
圖表 23 : P2P_1000_C_R

當 Super Seeder 頻寬為 500KBps 時，實驗結果如下圖：



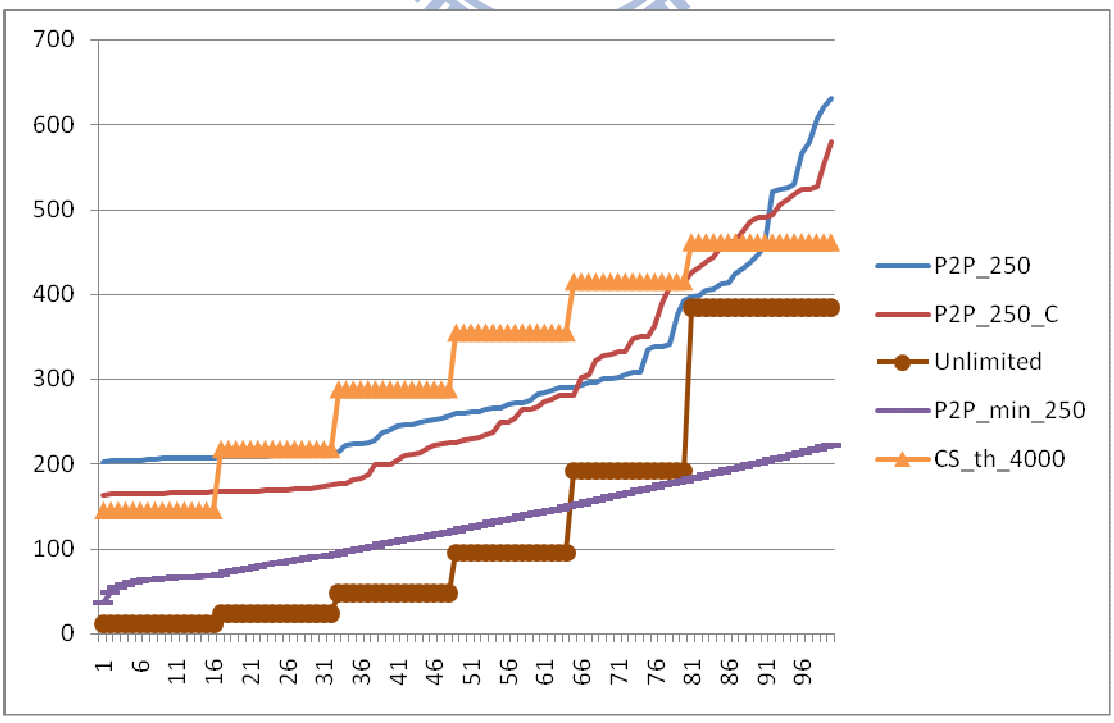
圖表 24 : P2P_500_C

由上圖可以看出，上傳頻寬較大的客戶端們，其下載速度有略為加快，而比值分佈如下圖：



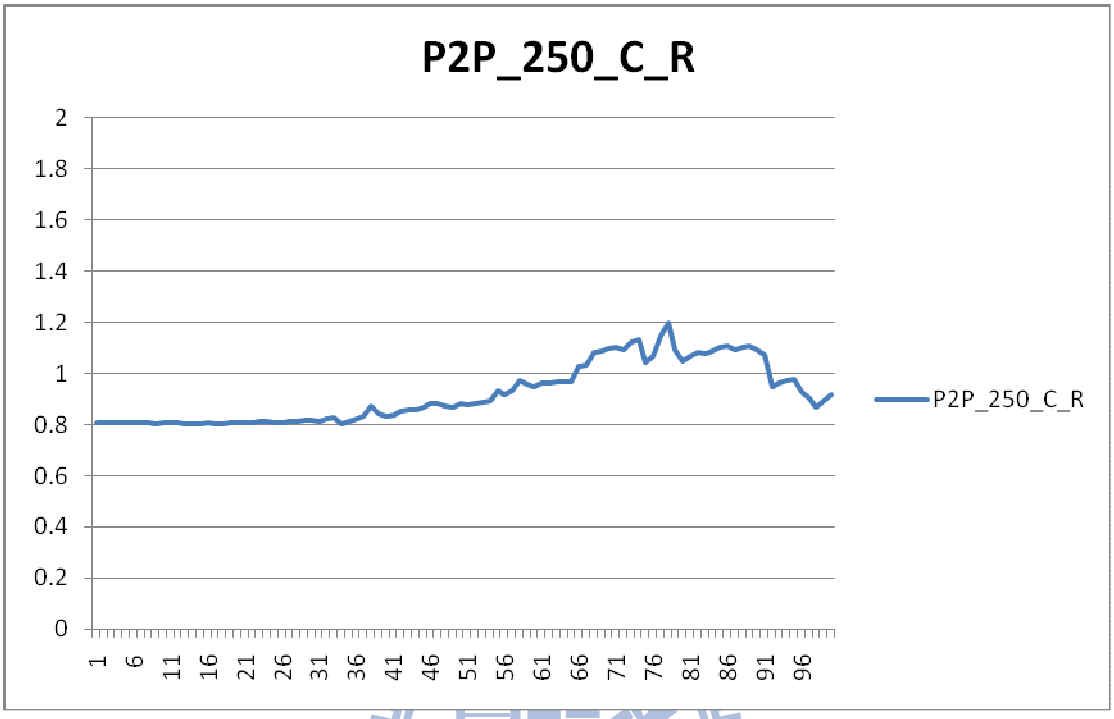
圖表 25 ： P2P_500_C_R

當 Super Seeder 頻寬為 250KBps 時，實驗結果如下圖：



圖表 26： P2P_250_C

由上圖可以看出，上傳頻寬較大的客戶端們，其下載速度有更為加快，而比值分佈如下圖：



圖表 27 ： P2P_250_C_R

由上圖可以看出，前面將近 50%的客戶端們，其下載時間約為原本的 0.8 倍，然而後面約有 25%的客戶端們，其平均下載時間約為原本的 1.1 倍。

4.3 Super Seeder 上傳策略改變

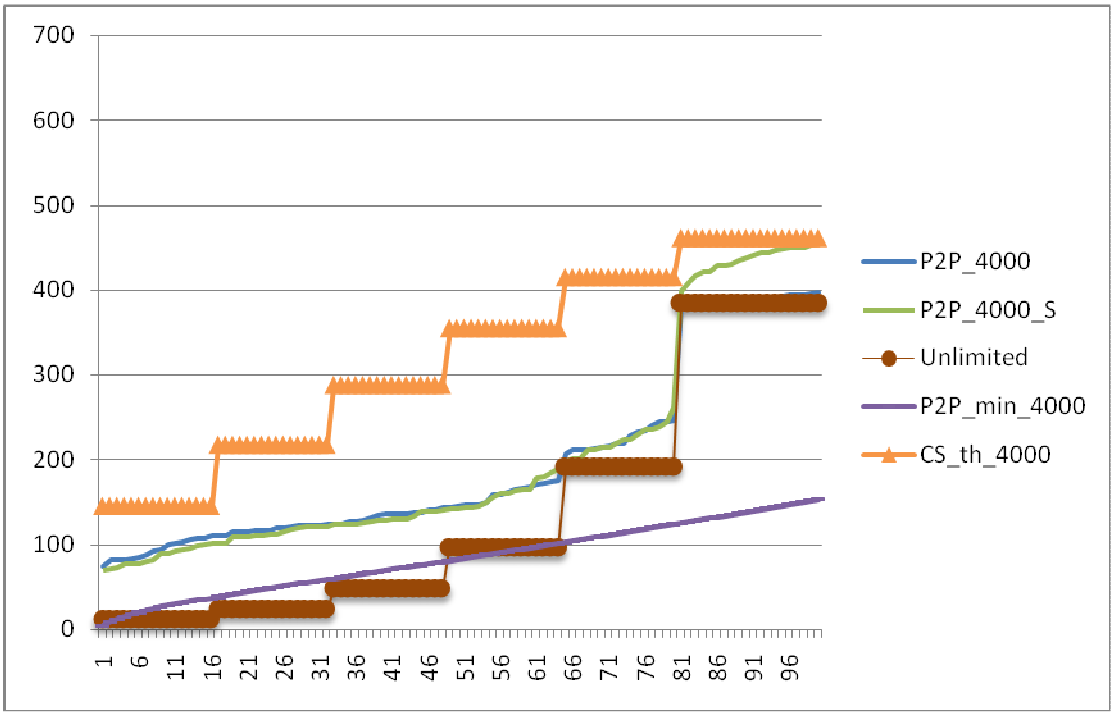
在 4.2 節中，我們嘗試了改變客戶端的上傳策略，希望藉由優先上傳給，上傳頻寬較大的客戶端們，來達到提高整體下載速度的效果，然而如前面實驗結果所顯示的，效果並不顯著，因此我們嘗試改變 Super Seeder 的上傳策略，希望藉此來提升整體效率，而改變的精神也是優先上傳給，上傳頻寬較大的客戶端群。

在此章節，我們將只對 Super Seeder 的上傳策略作改變，而原

本客戶端的上傳策略不作更動。原本 Super Seeder 公平上傳的精神改成優先選擇上傳頻寬較大的客戶端，但為了避免造成 starving 的現象，以及加速最後片段的下載，若客戶端要求的片段為一開始的部份片段，或最後的部份片段，則有較高的優先權。

接下來將展示 Super Seeder 在不同頻寬下的實驗結果，此章節的圖表將會和 4.2 節相同，觀察改變 Super Seeder 上傳策略後的下載速度。

當 Super Seeder 上傳頻寬為 4000KBps 時，發現兩者的下載速度有部份的差異，如下圖：

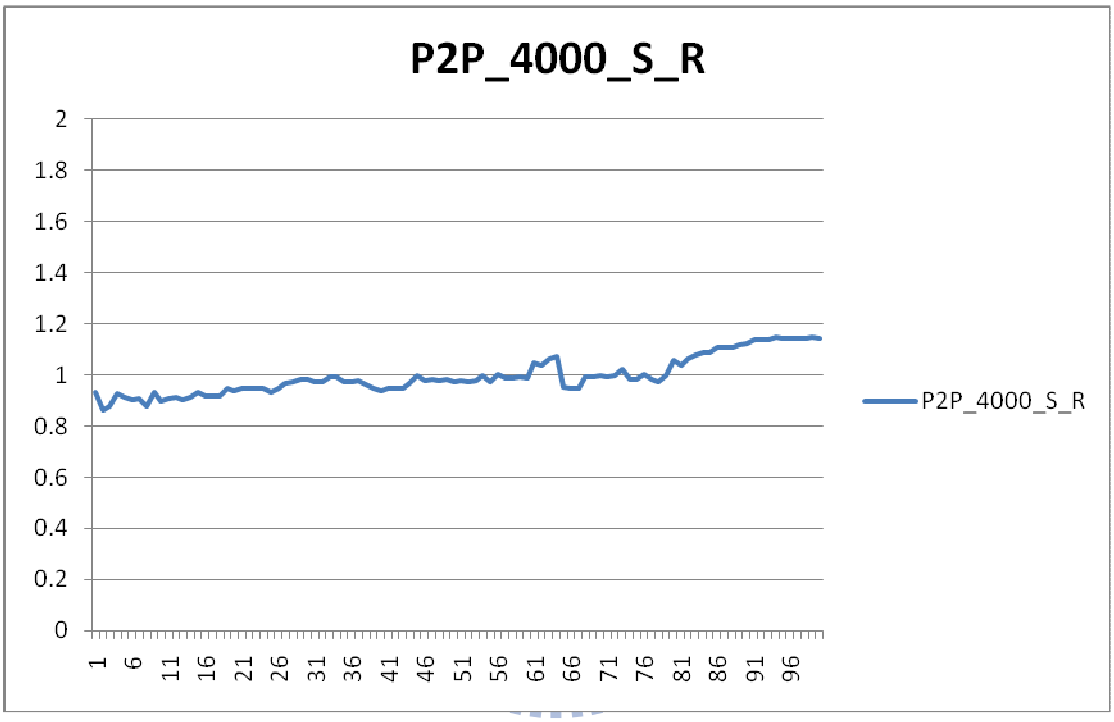


圖表 28 : P2P_4000_S

上圖中 P2P_4000 代表原本的下載速度，而 P2P_4000_S 則代表改變 Super Seeder 上傳策略時的下載速度，至於 Unlimited，CS_th_4000 與 P2P_min_4000 則為 4.1 節所提到的理論值，用來比較實驗結果，接下來類似的圖表也會以相同的方法來表達。

從上圖可以很明顯的看到，最後一個區間的客戶端群下載時間反

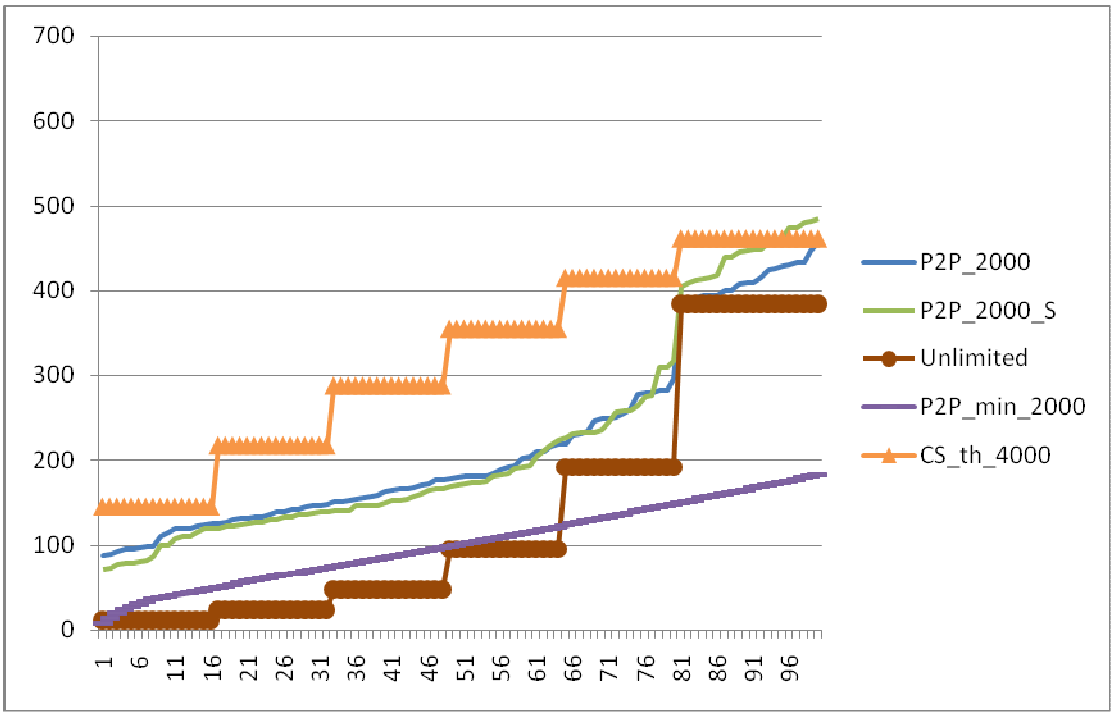
而拉長了，主要是因為當 Super Seeder 上傳頻寬為 4000KBps 時，其平均給每一個人的下載量約為 4000KBps/100 = 40KBps，因此在一開始下載時，已接近其下載速度的極限(48KBps)，然而很明顯的，Super Seeder 在改變上傳策略後，將不會平均分配頻寬給每個人，導至於最後部份的客戶端群的下載時間提高。接下來我們把上面的圖表改用比值的方式來呈現，如下圖：



圖表 29 : P2P_4000_S_R

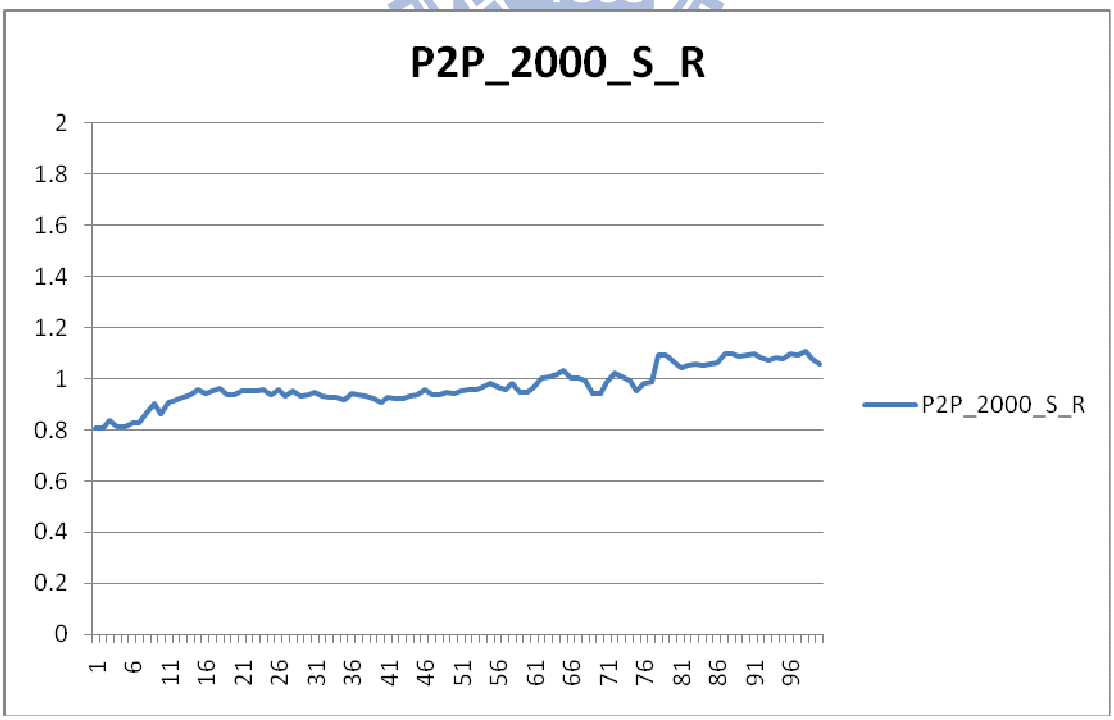
上圖中 P2P_4000_S_R 代表當改變 Super Seeder 上傳策略時，其下載速度除以原本的 P2P_4000 的下載速度後，所得到的比值，由上圖可看出最後部份的客戶端群下載時間，平均約為原本的 1.1 倍。

當 Super Seeder 上傳頻寬為 2000KBps 時，實驗結果如下圖：



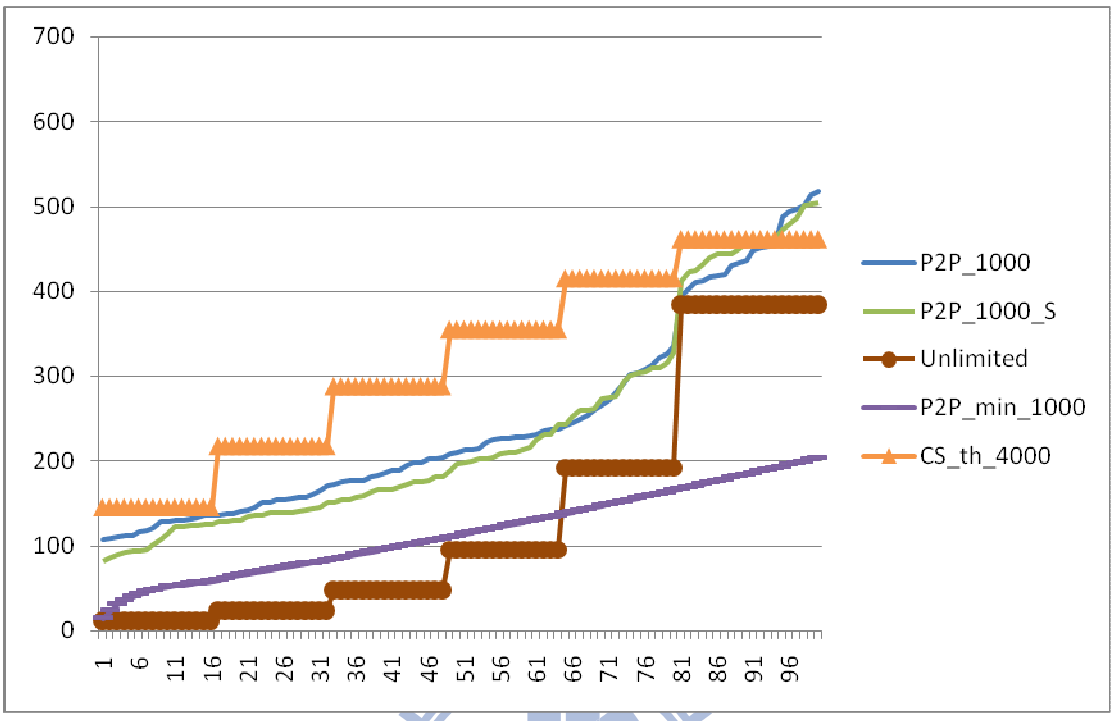
圖表 30：P2P_2000_S

由上圖可以看出，兩者的下載速度有漸為接近的趨式，而比值分佈如下圖：



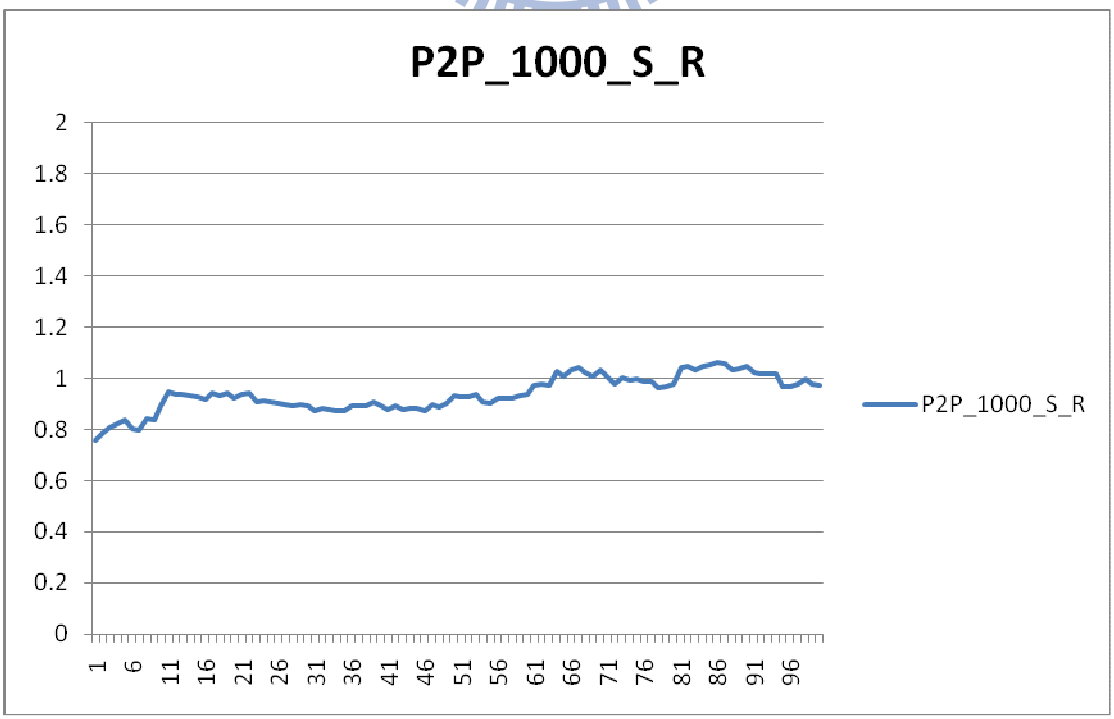
圖表 31：P2P_2000_S_R

當 Super Seeder 上傳頻寬為 1000KBps 時，實驗結果如下圖：



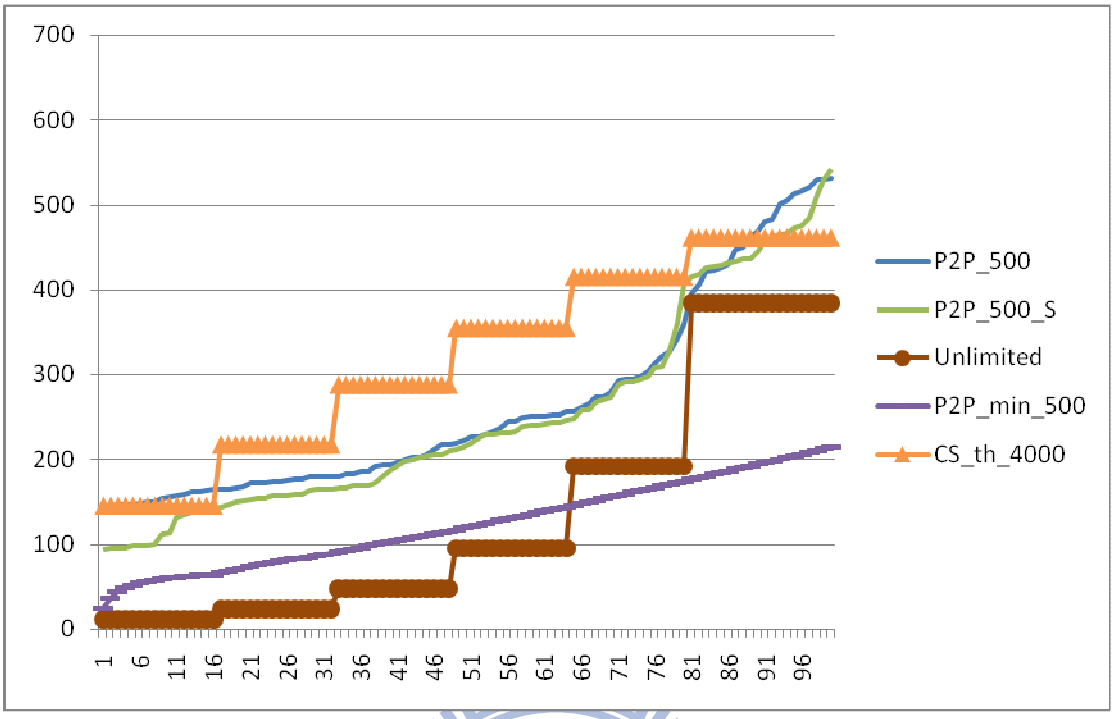
圖表 32 : P2P_1000_S

由上圖可以看出，兩者的下載速度非常接近且前面部份的客戶端群有稍微變快的趨勢，而比值分佈如下圖：



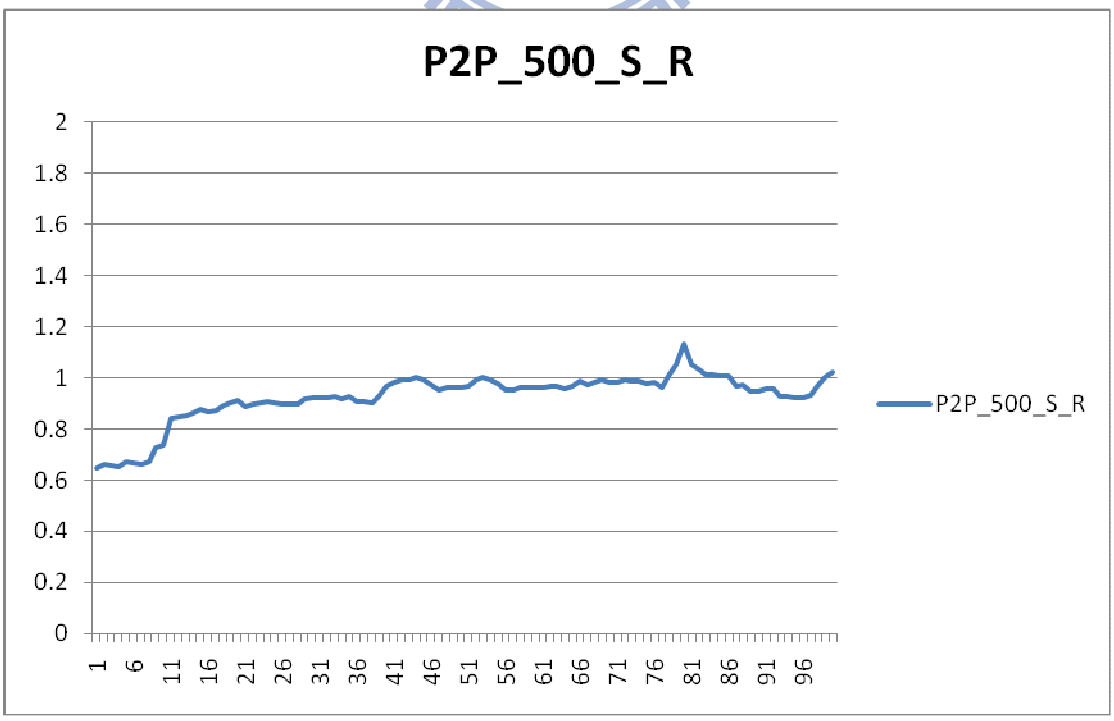
圖表 33 : P2P_1000_S_R

當 Super Seeder 上傳頻寬為 500KBps 時，實驗結果如下圖：



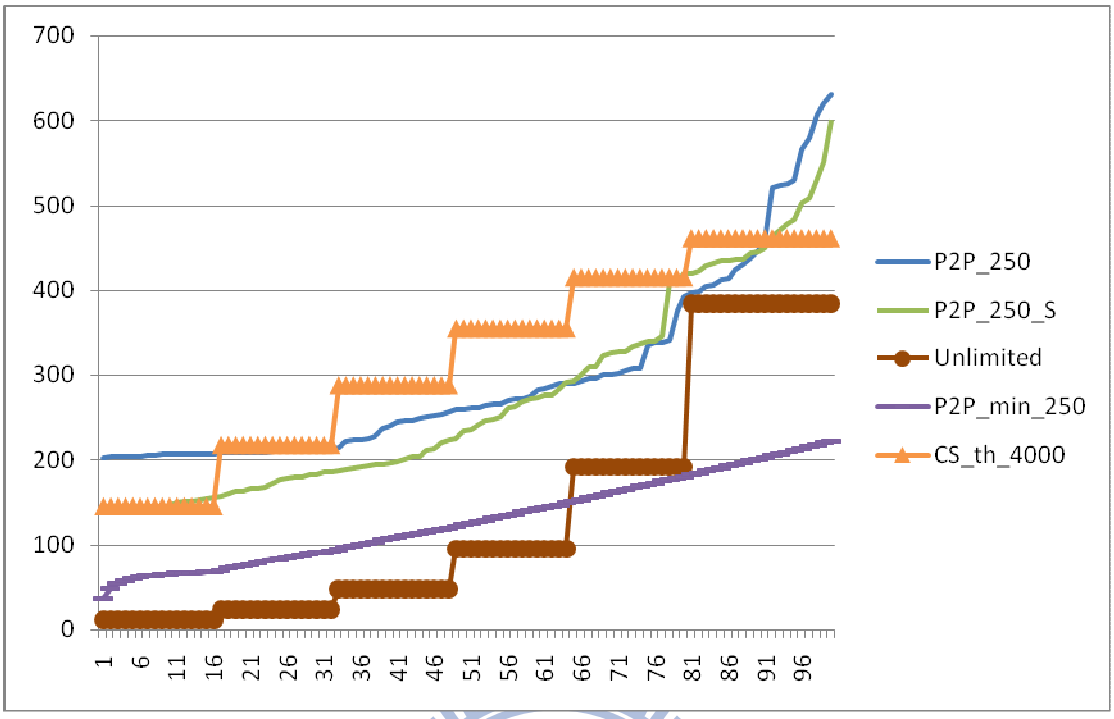
圖表 34：P2P_500_S

由上圖可以看出，下載速度有逐漸變快的趨勢，但整體差距仍然不夠明顯，而且主要為前面部份的客戶端群變快，而比值分佈如下圖：



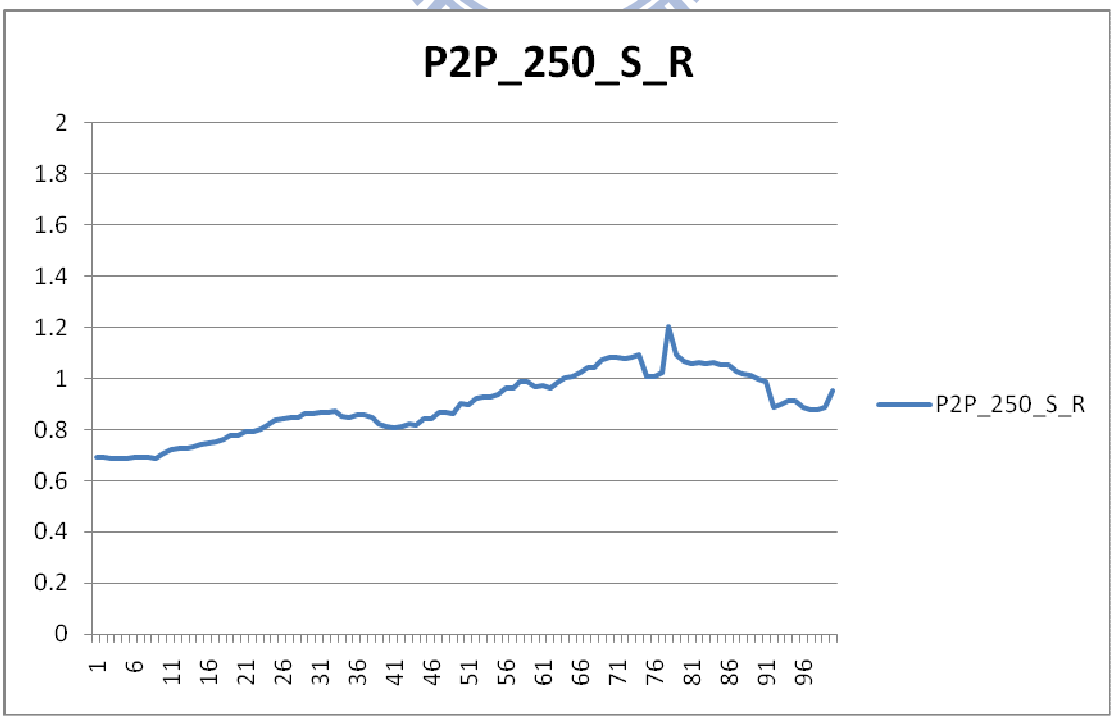
圖表 35：P2P_500_S_R

當 Super Seeder 頻寬為 250KBps 時，實驗結果如下圖：



圖表 36：P2P_250_S

由上圖可以看出，上傳頻寬較大的客戶端群，下載速度有比較明顯的加快，而比值分佈如下圖：



圖表 37：P2P_250_S_R

由上圖可以看出，前面加最後部份，將近 60%的客戶端們，其下載時間約為原本的 0.8 倍，然而中間後面約有 20%的客戶端們，其平均下載時間約為原本的 1.1 倍。

由 4.2 與 4.3 章節的實驗可以發現，當 Super Seeder 上傳頻寬很有限的情況下，客戶端或 Super Seeder 若優先上傳給，上傳頻寬較大的客戶端群，則可以提升部份的下載速度。

第五章、結論與未來展望

本論文提出了一個巨量發佈系統的客戶端實作方法，並比較公司在採用 MDS 系統與傳統 Client-Server 架構下，對玩家們下載速度的影響，並分析公司所可以節省的頻寬。之後透過改變 MDS Client 的上傳策略，及改變 Super Seeder 上傳策略，來顯示出當 Super Seeder 頻寬非常有限的情況下，優先服務頻寬高的玩家，能使得整體平均下載速度加快。

對於 MDS Client 及效能分析工具上，仍有許多部份是可以繼續加強的：

1. 將程式 porting 到 windows 環境上。
2. 透過改變上下載策略來達到更佳的整體下載效能。
3. 建立自動化模擬系統及設計良好的分析系統。

針對第一點，由於現在幾乎所有的線上遊戲玩家都使用 Windows 作業系統，因此我們透用 Cygwin 將 CTorrent 移植到 Windows 環境上 (透過夾帶 Cygwin 的 dll 檔)，然而，如果採用 Windows 原生函數的程式，則可以期許達到更好的效能。

關於第二點，從前一章的實驗結果可以發現，目前 MDS 系統整體的傳輸效能，離 P2P 的最快理論值仍有一段距離，然而這部份也有可能是因為在推導 P2P 最快理論值時，未考慮客戶端下載頻寬，而導致整體曲線評估過於樂觀。因此，接下來的系統發展可透過更精確的理論分析，來証明 BT 協定已達到很好的傳輸效果，或是透過改變上下載策略，來讓整體下載效能達到最佳化。

最後，對於第三點，由於目前所作的實驗都是靠同時操控多台電腦來作業，再加上環境設定的問題，使得實驗的手續相對煩雜。當整

體系統已經穩定到某種程度後，應當開發一個自動化的模擬系統，以及設計良好的分析系統，用來有效的驗証改變通訊策略後的整體下載效率。

參考文獻

[1] Shawn Fanning, Napster, Inc., Napster, http://www.napster.com/ , 1999 – 2001, 2004 – 2008.

[2] MetaMachine, eDonkey2000 (eDonkey, ed2k), http://www.edonkey2000.com , 2000~2005

[3] BitTorrent, Inc., *BitTorrent* http://bittorrent.com, 2002 – 2008.
Bram Cohen, "**Incentives Build Robustness in BitTorrent**", May 2003.

[4] *Rakesh Kumar, Keith Ross*, "**Optimal Peer-Assisted File Distribution:Single and Multi-Class Problems"**, Hot Topics in Web Systems and Technologies, 2006

[5] Arnaud Legout, I.N.R.I.A., Guillaume Urvoy-Keller and Pietro Michiardi, Institut Eurecom Sophia Antipolis, France, Technical Report, "**Understanding BitTorrent : An Experimental Perspective**" , November 2005

[6] M. Izal, G. Urvoy-Keller, E. W. Biersack, P. A. Fellber, A. Al Hanmra, and L. Garces-Erice, "**Dessecting BitTorrent: Five Months in a Torrent's Lifetime**" , 2004

[7] Microsoft Research, "**Analyzing and Improving BitTorrent Performance**" , 2005

[8] *Che-Yi Lin*, "**The Study of Massive Deployment System Based on P2P Technology – The Servers and System**

[9] Enhanced CTorrent
http://www. rahul.net/dholmes/ctorrent/

[10] Enhanced CTorrent User's Guide
http://www.rahul.net/dholmes/ctorrent/userguide.html

[11] BitTorrent Specification
http://wiki.theory.org/BitTorrentSpecification, February 2008

[12] *K. Egevang, P. Francis*, RFC 1631, "**The IP Network Address Translator (NAT)**".

[13] 群想網路科技, "客戶端 NAT 與頻寬分析", 內部文件, August 2008.

[14] *Yi-Chan Shan*, "**Performance Analysis of P2P File Distribution**", private communication, 2008.