國立政治大學資訊科學系
Department of Computer Science
National Chengchi University

碩士論文

Master's Thesis

適用於 P2P 檔案分享系統傳輸協定之設計

A UDP-Based Protocol for Distributed P2P File Sharing

研 究 生：許弘奇

指導教授：連耀南

中華民國九十四年十月

October 2005

適用於 P2P 檔案分享系統傳輸協定之設計

摘要

Peer-to-Peer(P2P)架構讓社群內的使用者收集分散在網路各處之資源，使參與者得到原本無法負擔之運算資源，其中最為風行的 P2P 系統當屬 P2P 檔案分享系統。P2P 檔案分享系統之架構可分為集中式及分散式，而分散式架構又可細分為結構化及非結構化兩種。採用分散式且非結構化之 BitTorrent-like 架構，因其可擴張性較佳而廣為風行。BitTorrent 傳輸層協定採用 TCP 協定。在經驗中發現，BitTorrent-like 架構在非對稱網路之下雖擁有寬裕的下行頻寬，但是其頻寬使用率卻不高，我們分析其成因如下：(1) Fractional Upward Bandwidth 、(2)Blockage of Acknowledge 與 (3)Long Physical Paths 等，其中 Blockage of ACK 問題，現今尚未有研究學者提出完整解決方案。本研究之目的，即要針對此 Blockage of ACK 的問題，提出一可行之改進措施。我們改良網路協定中的傳輸層 (Transport Layer) 協定以提昇 P2P 檔案分享系統之效能，改用 UDP 作為傳輸層協定，並在應用層加入自動重建遺失之封包(Packet Loss Recovery)、決定基本傳輸單位大小(Segment Size Determination) 及決定資料傳送速率(Date Rate Determination)等機制，以彌補 UDP 之缺陷。文中並提供了傳輸協定運作時所需參數的估計法，並且也與其它傳輸協定做效能之比較。實驗結果發現，我們設計的傳輸協定確可改善 P2P 檔案共享系統的運作效能。

A UDP-Based Protocol for Distributed P2P File Sharing

Abstract

Peer-to-Peer (P2P) architectures let participants gather resources form network and make participants acquire more computation resources than they could offer. One of the most prominent P2P systems is P2P file sharing system. P2P file sharing system could be classified into to 2 categories: centralized and decentralized model. BitTorrent-like model is a kind of decentralized and unstructured model. BT-like model is quit popular nowadays due to its scalability. Unfortunately, BitTorrent-like model has several shortcomings on performance over asymmetric networks, because of these problems: fractional upward bandwidth, blockage of acknowledgement and long physical paths. There are no complete solutions for blockage of ackownlegement. We propose a new UDP-based protocol to alleviate this problem. UDP protocol lacks for data-integraty and can't determine transmission rate by its own. To form a complete solution, we have to design some co-operating mechanisms for UDP, such as packet loss recovery, segment size determination and data rate determination mechanism etc. Experiments have shown that our proposed protocol has good improvement on performance. We hope that our protocol would help P2P file sharing participants have performance gain.

目錄

第一章

簡介

P2P 架構允許網際網路上一社群之使用者將其資源收集起來 (Content, storage, network bandwidth, disk space, CPU time etc.)[14]，故社群內之參與者可存取到較大的資料儲存空間、舉行多人同時參與的視訊會議、進行較複雜的搜尋和計算等等，諸如此類。P2P 系統可讓每個參與者享受到原本無法負擔的資源[6]。網際網路上，除了 P2P 架構外，另一常見之服務架構為 Grid 架構[14]。Grid 與 P2P 架構間之差異在於 Grid 架構需伺服器(Server)角色來提供服務，但 P2P 架構不需伺服器的角色，每位參與者皆可為服務提供者[13]。

P2P 架構最廣為風行的應用當屬檔案共享系統 (File Sharing System)，如 Napster[30]或 BitTorrent[2]。P2P 檔案共享系統內，參與之角色分別為：(1)資料提供者(Data Source Provider)及(2)資料下載者(Downloader)[17]；而運作方式則分集中式及分散式兩種架構[16]：

(1)集中式架構 (如 Napster-like Model)，有一共同資料來源，供其他人抓取檔案，當下載者太多時，系統無法負荷。

(2)分散式架構 (如 BitTorrent-like Model)，下載資料者亦可幫忙提供已下載之資料供他人下載，此種架構可供較多人同時下載。

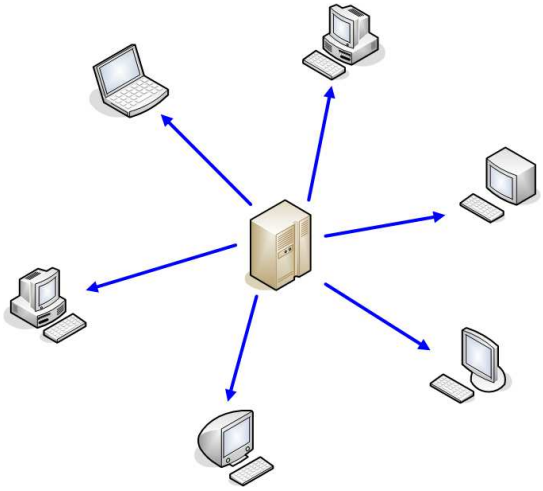


圖 1.1：集中式架構

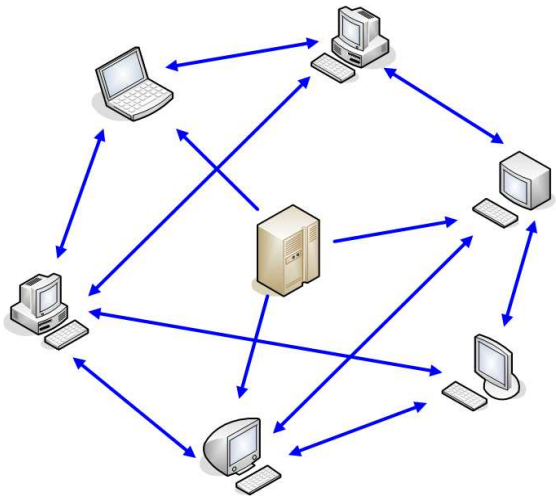


圖 1.2：分散式架構

分散式架構項下又可細分[16]：

(1)結構化之 P2P 檔案共享系統，其檔案來源路徑已將網路拓樸位置納入考量，可避免路徑重疊的問題。

(2) 非結構化之 P2P 檔案共享系統，其檔案來源路徑未將網路拓樸納入考量，故其資料傳輸路徑有可能嚴重重疊，因而浪費大量網路頻寬。

由於 BitTorrent-like Model[2]為目前最風行之 P2P 檔案分享系統架構，此架

構利用多點對多點之傳輸方式，具有參與之下載者愈多，其下載速度不會大幅降低之優點。我們的研究，將著重於探討分散式非結構化之 BitTorrent-like Model 之相關議題。

1.1 BitTorrent-like Model

1.1.1 BitTorrent 之角色

BitTorrent[2]系統運作時，參與者可分下列四種角色：

(1)資料提供者(Seeder)：原始完整檔案提供者。

(2)資料下載者(Downloader)：欲取得該檔案之參與者。

(3) Tracker：BitTorrent 檔案分享系統之中央控管角色，協助下載者尋找所需之檔案。

(4)網頁伺服器(Web server)：公佈並提供欲下載檔案之相關資訊。

1.1.2 BitTorrent 之特色

在[2][5]幾篇文章中，提到 BitTorrent 的幾項特色如下：它是一種 P2P 檔案分享協定，運作架構採用分散式且非結構化之模式。整個系統之啟動，需有一欲提供檔案之檔案提供者，此檔案提供者利用 BitTorrent Client 軟體將檔案切割成多個檔案片段(Fragment 或 Piece)，並公佈此檔案之相關資訊至某網頁伺服器上。若有檔案下載者欲下載該檔案，可至該網頁取得相關資訊之後，才可進行下載。BitTorrent 協定最特殊的地方，在於進行檔案分享時，每個下載者在下載同時亦會開放已完成之檔案片段供其它下載者抓取，而且每個下載者在下載檔案時，可從不同之地點下載。(同一檔案片段同時有許多地點可供下載；下載者可從不同地點下載同一檔案片段。)此檔案分享協定具有可擴張性(scalability)之優點[13]：

參與者愈多，其下載者之下載速度不會大幅降低。

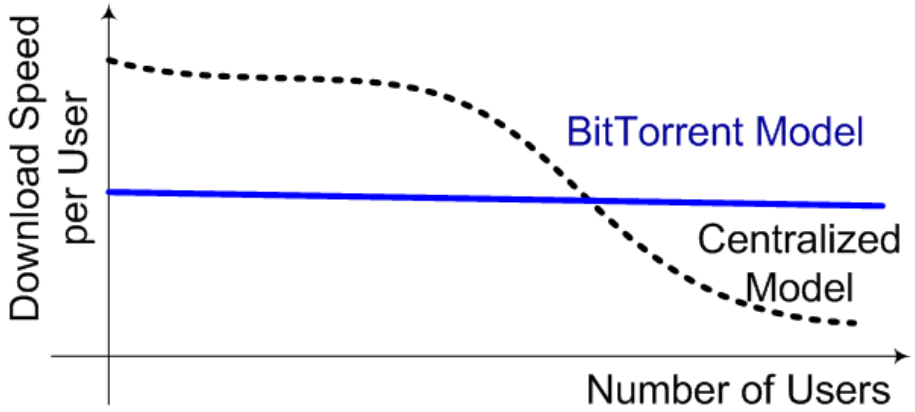


圖 1. 3：下載速度與參與人數關係圖

1.1.3 BitTorrent 之運作機制

無論是資料的提供者或者是下載者，都需安裝 BitTorrent Client 軟體。BitTorrent 協定的運作方式可分為三個項目：檔案提供、檔案公佈、及檔案下載[2]。

(1)檔案提供：BitTorrent 系統之運作，最初需有一原始完整檔案提供者，稱為 Seed，檔案之分享必須先有一個 Seed 方能開始運作。此最初之原始檔案提供者利用 BitTorrent Client 程式針對欲分享之檔案建立 .torrent 檔。在建 .torrent 之過程中，並同時需指定「Tracker」之 URL。

(2)檔案公佈：原始檔案提供者需將 .torrent 檔公佈至某網頁伺服器上。此 .torrent 檔中，除了有 Tracker URL 位置之外，亦包含被下載檔案的相關資訊，如檔名、大小、Signature 等相關訊息。

(3)檔案下載：任一下載者欲下載檔案時，需先經由 web 抓取 .torrent 檔才

可進行下載。下載者用 BitTorrent Client 程式開啟此.torrent 檔，即可進行檔案下載之動作。其中 BitTorrent 進行檔案下載時，會經由「Tracker」來找到其他下載者。

此外，Bran Cohem[2]為 BitTorrent 定義了兩個基本檔案單位：Piece (Fragment)及 Sub-piece (Sub-fragment)。一 Piece 為 1/4 Mbyte，為 Tracker 公布給其它下載者之檔案管理基本單位；一 Sub-piece 為 16Kbytes，是利用 Pipelining 方式提昇 Piece 傳輸速度之單位。BitTorrent 之 Pipelining 之目的為提昇網路頻寬使用率。因為 BitTorrent 傳輸層採用 TCP 協定[21]，TCP 協定在傳送資料時，接收端在收到傳送端之資料後，須對下一個資料提出請求；等到接收端之請求送達傳送端時，傳送端才繼續傳送下一資料。此一機制，導致傳送端偶需暫停以等待下一個資料請求之到達，因此產生時間空檔。BitTorrent 為改進此一 TCP 效能問題，提出 Sub-piece 之 Pipelining 機制，根據 Pipelining 機制將 Piece 切成多個 Sub-piece，同時建立多個連線(Connection)進行資料傳送，接收端隨時皆在對新資料提出請求，避免時間空檔、提昇網路頻寬使用率。

在[2]中，作者亦說明了要如何選取要被下載的 Piece，以及解釋原作者精心設計之 Choking Algorithm 如何協調各個下載者的上下行速度，以達到柏雷拖最適效率(Pareto Efficiency)的檔案傳輸關係等。

最近，BitTorrent 發表一個新的 beta 版，稱為「trackerless」[15]。原始資料提供者在建 .torrent 檔時，可以自行選擇是否要使用 trackerless 的選項。若此 .torrent 檔選用了 trackless 的選項後，下載者利用此 .torrent 進行下載時，不需經過 Tracker 即可下載檔案。不過，雖然它稱之為 trackerless，但實際上之運做是每個下載之 client 都扮演 lightweight tracker 之角色，故可協助其他下載者找到彼此。關於 Tracker 與 Trackerless 之比較如下：

(1)使用有專屬 tracker 之選項時，因可以收集到系統整體運作之統計數據，故可提供控制時之參考資訊，而增加系統整體運作之可靠度。

(2)使用 trackerless 之選項時，不需要一專屬之 Tracker，但無法對系統運作提供一整理管理參考用之資訊。

1.2 TCP 簡介

1.2.1 TCP 之特色

TCP 協定[21]是端對端(end-to-end)之傳輸層協定(Transport layer protocol)，一個傳送端，一個接收端。協定提供可靠(data integrity)、依序(in-order)之保證。因為協定為 connection-oriented，在資料傳輸之前需先建立連線狀態。採用 Positive ACK 的方式，並且經由 ACK 驅動傳送端送出新的封包(self-clocking)。它是有流量控管機制(flow control)，傳送端之資料速率不會超過接收端所能負載之限度。經由擁塞窗框(congestion window)之調整來控制傳輸速率。而擁塞之信號是封包遺失或封包延遲。

1.2.2 TCP 之擁塞控管

TCP 是利用 Window Size 來調節資料傳輸速率。以封包遺失或延遲當作網路擁塞的指標。資料傳輸中若有封包遺失或延遲，TCP 就會啟動擁塞控制機制快速降低資料傳輸速率。其運作共分兩個階段：Slow Start 與 Congestion Avoidance。

(1)Slow Start 階段(CWND < Threshold)：TCP 在資料開始傳輸時，會利用 Slow Start 機制來探測目前網路可承載之頻寬。當 connection 建立以後，CWND 大小以指數的速度成長，直至超過 Treshold 或封包遺失產生為止。

(2)Congestion Avoidance 階段(CWND > Threshold)：在擁塞窗框超過之後，

再利用 AIMD (Additive Increase Multiplicative Decrease)的方式進行擁塞控管。此機制中，是以封包遺失當作網路擁塞的指標。資料傳輸中若有封包遺失，TCP 就會啟動擁塞控制機制而會快速降低資料傳輸速率。下圖 1.4，即為 TCP 擁塞控管機制之示意圖。

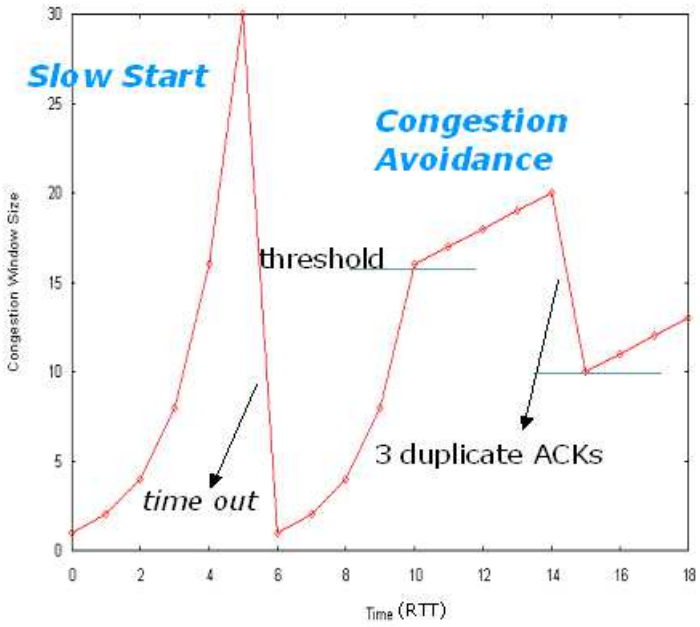


圖 1. 4 : TCP 擁塞控管示意圖

1.3 P2P 檔案共享系統之效能議題

1.3.1 P2P 檔案共享系統的效能缺陷

雖然 P2P 檔案共享系統被大量使用，但是關於網路協定對 P2P 傳輸效能所造成的影響研究卻一直未受廣泛重視。由經驗中發現，在上行頻寬窄下行頻寬大的非對稱網路之下(如 ADSL)使用 BitTorrent 時，整體資料傳輸的效能並不佳。我們提出一個疑問：「當擁有寬鬆的下行頻寬時，為何下載速度很慢？」我們即

針對此問題進行分析。

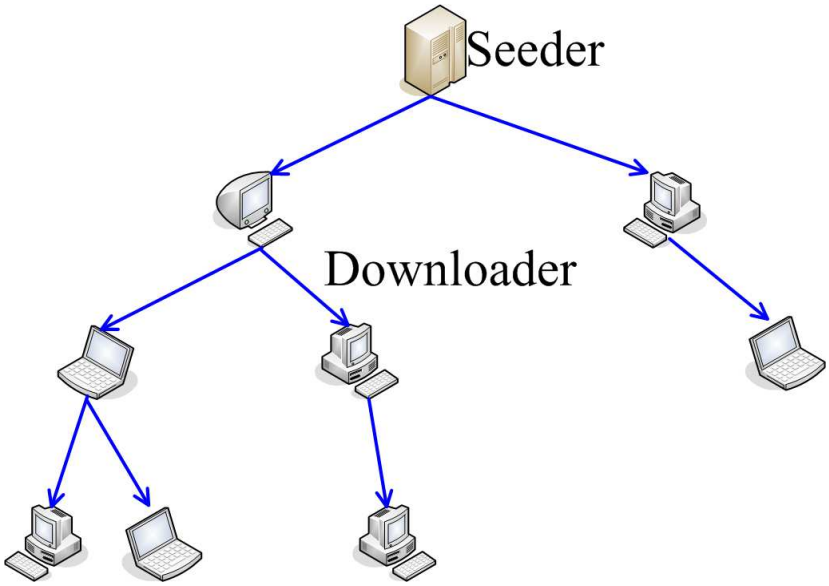
1.3.2 P2P 檔案共享系統效能問題之分析

關於 P2P 檔案分享系統效能不彰問題的成因可以分析如下[2]：

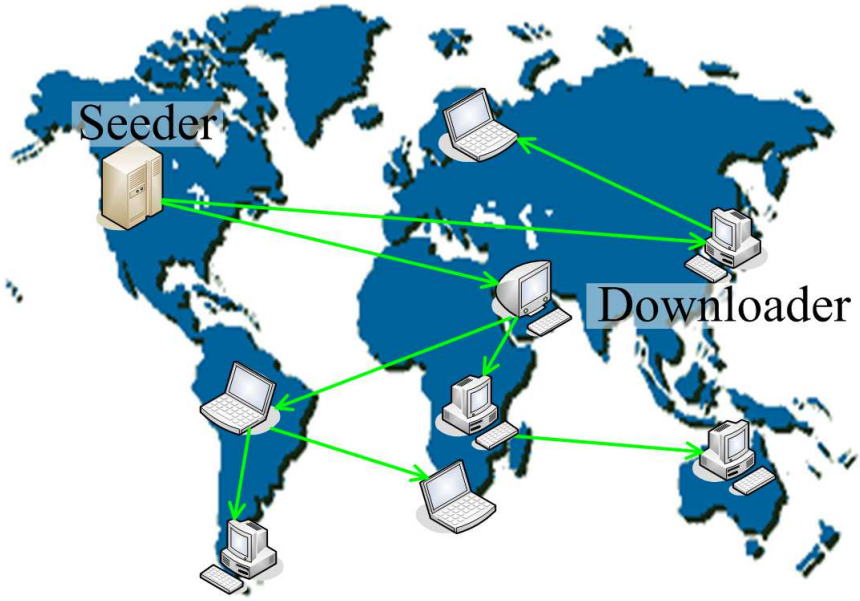
(1)Fractional Upward Bandwidth (FUB) 的問題：一部電腦內的檔案分段同時被多個下載點同時下載，但因 ADSL 上傳的頻寬窄，導致多個上傳訊務流要共用一個較窄的頻寬。每個上傳訊務流可獲得之上行頻寬因此而更形不足。

(2)Blockage of Acknowledge (BoA) 的問題：TCP 協定在運作時，接收方在收到資料之後，必須回傳一個 ACK，但當運用 BitTorrent 下載檔案時，作為 TCP 接收者的下載者亦正在上傳資料給其他使用者，較窄之上行頻寬因有上傳的訊務流而顯得擁塞，使得 ACK 無法順利回傳。ACK 在路徑上被丟棄或是逾時，導致 TCP 誤以為網路擁塞，因而啟動擁塞控制機制，主動降低傳送速度。

為了讓問題容易分析，我們可將檔案資料每個下載者所形成的關係想像如圖 1.4(a)：同一個檔案片段(File Fragment)以 Seeder 為樹根(Root)，其資料上傳者和資料下載者間的關係會形成一個階層架構，我們稱之為檔案片段樹(Fragment Tree)。但是在檔案片段樹之一個鏈結，實際上為兩部電腦端對端之路徑，其距離有可能很長。圖 1.4(b)即為此 Fragment Tree 實際上之可能狀況。



(a)



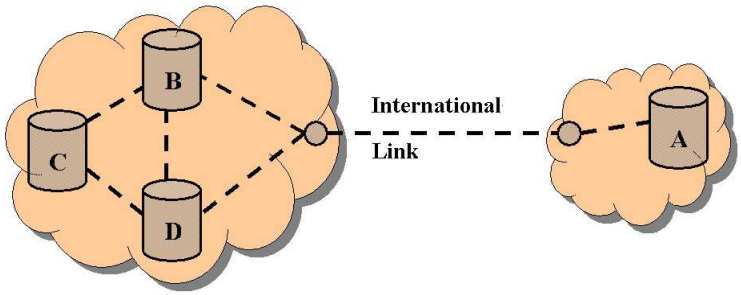
(b)

圖 1. 5 : Fragment Tree (a)Logical Fragment Tree (b)Physical Fragment Tree

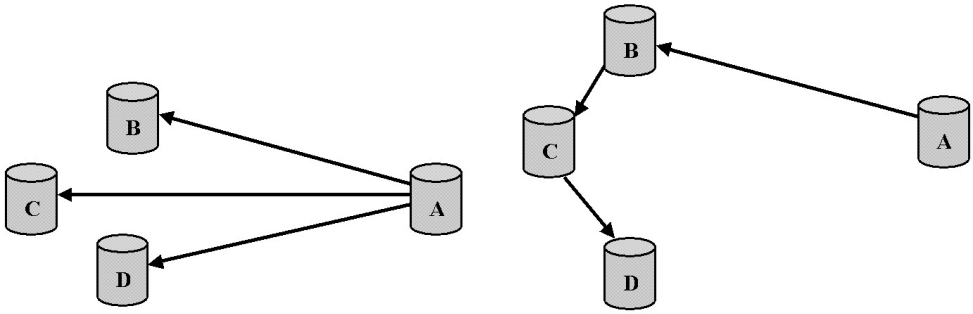
由於檔案片段樹觀察可以看出幾個問題：

(1)Long Physical Paths：在檔案片段樹中兩部電腦間的鏈結(link)事實上是一

個路徑，下載者下載檔案時並未將路徑長短納入考量，很可能會從遠方下載檔案，下載路徑太長浪費網路資源，且各個下載路徑之間此不免有重疊之處。[5]提出，如果能盡量縮短路徑，減少重覆，必能降低頻寬之浪費。



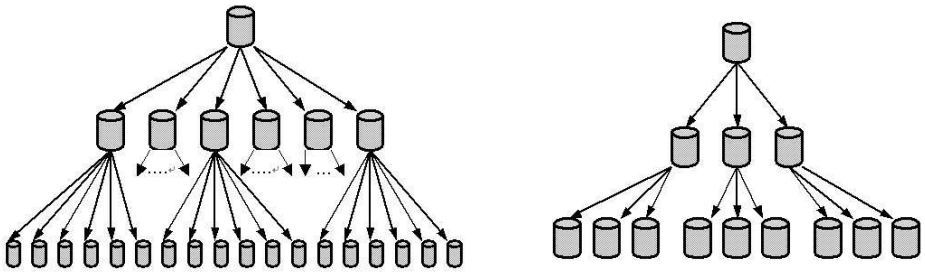
(a)



(b) (c)

圖 1. 6 : (a)原來的網路拓樸 (b)下載路徑有重疊之處 (c)避免重疊之路徑

(2)Bushy Tree: 下載者讓太多其他使用者下載自己已經下載完成的檔案片段，以致產生 FUB 與 BoA 的問題。在[5]提出，可將之改成分支度較小的 Slim Tree 來讓每個人分享出的資料流數目控制在合理範圍內[5]，即可有效控制 Bushy Tree 之負面影響。



(a) (b)

圖 1. 7 : (a)Bushy Tree (b)Slim Tree

(3)其它 BitTorrent-like Model 議題，如資源尋找等等。

1.6 研究動機與目的(Motivation and Research Objective)

BitTorrent-like 因為有目前最為風行且可擴張性佳等優點，其運作效能議題已逐漸受到學界的重視[9][10]。我們在經驗中發現，對非對稱網路(如 ADSL)的下載者而言，其下行頻寬大然而其頻寬使用率卻不高。BitTorrent-like 系統在非對稱網路之下有效能無法提昇之問題。由我們之前的分析，可以得知，這是受 BoA (Blockage of ACK) 與 FUB (Fractional Upward Bandwidth)、Bushy Tree、Long Physical Paths 等機制的影響。其中 BoA 問題現今尚未有研究學者提出完整可行的解決方案，本研究的目的，即針對 BoA 效能問題提出可行之改進措施。

1.7 解決方案(Solution Approaches)

在第三章中，我們提出一個 UDP-Based Approach 來改進 BoA 的問題。UDP-Based Approach 內有三項特殊機制，分別是 Packet Loss Recovery、Segment Size Determination 與 Adaptive UDP Mechanism。

(1)Packet Loss Recovery (自動重建遺失之封包)： 為避免封包遺失引發太多

的資料重傳，我們加入自動重建遺失封包之機制。

(2)Segment Size Determination (傳送單位之計算)：計算檔案片段(Fragment)中，最佳的基本傳輸單位(Segment)大小，使得總傳輸量為最低。

(3)Adaptive UDP Mechanism (調節式 UDP 機制)：彌補 UDP 沒有自動調整傳輸速率的問題。我們希望能根據網路狀態，將資料傳送速度調整到最適的傳輸速率。

我們的 UDP-Based Approach 即以上述三項機制為核心，希望能有效解決 BoA 導致之效能問題。

1.8 論文組織架構

本文第一章分析 BitTorrent-like Model 在採用 TCP 協定時於非對稱網路下所遭遇到之各種效能議題；第二章我們回顧非對稱網路之相關文獻以及其他研究學者對於 BitTorrent-like Model 效能議題的一些看法；第三章介紹本研究所提出之「UDP 為基礎之協定」之細部設計及有限狀態機運作流程圖；第四章展示本研究之協定參數估算結果並以模擬方法評估我們提出協定的運作效能；第五章為本文結論與未來展望。

第二章

相關研究

為了研究BitTorrent之效能議題及其在非對稱網路下之效能影響，我們以下就BitTorrent-like之效能議題其在非對稱網路下之效能影響，回顧近幾年之相關研究。

2.1 BitTorrent-like 效能議題

2.1.1 對 BitTorrent 之效能觀察

對 BitTorrent 之效能觀察研究，有 M. Izal 等人在 2004 年發表之觀察研究[9]，他們以 BitTorrent 檔案共享系統散佈 Linux Redhat 9 作業系統之光碟映像檔(共 1.77GB)，並針對此散佈五個月期間之 tracker 及 client 紀錄檔進行研究分析。研究人員得到下列結論：

(1)BitTorrent 之運做模式有相當之可擴張性。

(2)每個下載者之平均下載速率高於 500kb/s。

(3)下載速度和上傳速度為正相關。

(3)下載者取得部份檔案片段後，即可將此片段供其它下載者抓取。

(4)下載者完成下載之後，平均還會保持連線 6.5 小時。

2.1.2 對 BitTorrent 之效能分析

以分析及數學模型來探討 BitTorrent 之效能有 D. Qiu and R. Srikant. "Modeling and Performance Analysis of BitTorrent-like Peer-to-Peer Networks," [10]。文中利用數學分析來瞭解 BitTorrent 之效能。利用 Fluid Model 進行分析。文中的結論：

(1)這個系統的規模可擴性相當好

(2)檔案分享亦很有效率。

此 Model 的缺點為，未將幾項重要 BitTorrent 參數納入考量，如 node degree、maximum concurrent uploads、environmental conditions(如 seed bandwidth)等，此等幾項參數對 BitTorrent 實際運做時之效能會有相當程度的影響，缺漏這幾項參數為此分析模型之一大缺憾。

2.2 非對稱網路下之 TCP 問題

2.2.1 非對稱網路下 TCP 問題之回顧

討論非對稱網路下 TCP 問題有 H. Balakrishnan and V. N. Padmanabhan, “How Network Asymmetry Affect TCP”[1]討論基於回饋之傳輸協定(feedback-based)，如 TCP 於非對稱網路下之效能影響。

文中，對非對稱有如下三種分類：

(1)頻寬的非對稱。

(2)介質存取的非對稱。

(3)資料遺失的非對稱。

作者對非對稱網路之處理，必需對上行頻寬的使用進行管理，可採用之手法整理如下：

(1)TCP Header Compression。

(2)ACK Filtering。

(3)ACK Congestion Control。

(4)ACKs First Scheduling。

但是若因此導致 ACK 的頻率變低，會破壞 TCP 協定中原有的 Self-clocking 機制。所以尚需加上 ACK Reconstruction 等機制的幫助，才可成為一個較佳的解決方案，但是卻需額外的網路層(layer 3)設備的幫助。

2.2.2 非對稱網路下 TCP 問題之解法

而關於非對稱網路的傳輸問題，在[4]中提到，各種 TCP 版本中，廣受大家討論的有 TCP Tahoe、TCP Reno、TCP Vegas 等等這幾種[27][28]。其中，TCP Vegas 經由觀測 Round Trip Time (RTT)時間的變化，來瞭解瓶頸鏈結(Bottleneck Link)的佇隊長度(Queue Length)變化情形，依此控制傳輸速率而得到相當不錯的效能表現。但是，由於 TCP Vegas 不能分辨在非對稱網路之下，因傳輸方向不同所產生的負面影響，而導致整個效能大為降低[29]。此文中提出一個新的 TCP Formosa 協定。此協定中為了解決非對稱網路問題，設計了一個 Cumulative ACK 的機制，減少了原來傳輸時 ACK 的數量，避免非對稱網路之下因為 ACK 遺失所導致的

影響 ，進而在一定程度上解決了非對稱網路 TCP 的效能問題，獲得了不錯的成果。

2.3 評論

上述針對非對稱網路處理之方法皆是直接修改 TCP 協定。改變 TCP 茲事體大，協定更改幅度太大，影響層面廣，不易被接受。現有之使用者不易為了解決非對稱網路產生之問題即採用一個新版的 TCP 協定。

第三章

以 UDP 為基礎之解決方案（UDP-Based Approach）

有數個方案可以解決 BoA 所導致的效能降低問題：

第一個方案是增加 TCP ACK 逾時的時間。但此法有一副作用，因為 TCP 是利用逾時的時間決定一個封包是否遺失，增加 TCP ACK 逾時的時間，會導致 TCP 對真正網路擁塞的反應時間拉長，而不能即時處理網路擁塞(亦即立刻降低傳送速率)。

第二個方案是令 TCP 之接受端重覆傳送 ACK，以增加 ACK 之存活率。但是此法會在非對稱網路上的上行端增加多餘的 ACK 訊務量，浪費了寶貴的上行頻寬。另外，此法需對 TCP 協定作一定幅度之修改。

本文提出第三方案，以 UDP 為基礎的方式(UDP-Based Approach)解決。此法，假設傳送者握有實際可用頻寬(Actual Available Bandwidth)之大小。使用 UDP 為基礎方式(UDP-Based Approach)進行資料傳送，因接收端不需要對接收到的封包回送 ACK，故可以避免 BoA 問題，傳送端不會無謂的降低傳送速度。但 UDP 協定本身有兩個問題必須解決，(1)無法確保資料的完整性。(2)無法自動決定資料傳送速率，因此我們在應用層(Application Layer)上加上確保資料完整性及決定傳送速率的機制。此協定之特點是以 UDP 協定為基礎，並在應用層加入確保資料完整性與自動決定資料傳送速率之機制。在設計上，需考慮下列項目：

(1)發生封包遺失時，會導致傳送端資料之重傳，故協定須提供自動重建遺失封包之機制。

(2)因基本傳輸單位大小會影響傳送成功率，故要設計一傳輸單位大小之計算功能。

(3)為提供自動決定資料傳送速率之機制，故需經由量測可用頻寬，來決定傳送速率。

此外，協定預設之採用者為 BitTorrent 架構下之參與者，其中傳送端與接收端(sender & receiver)皆需採用我們提出之協定。所設計之有限狀態機如圖 3.1。

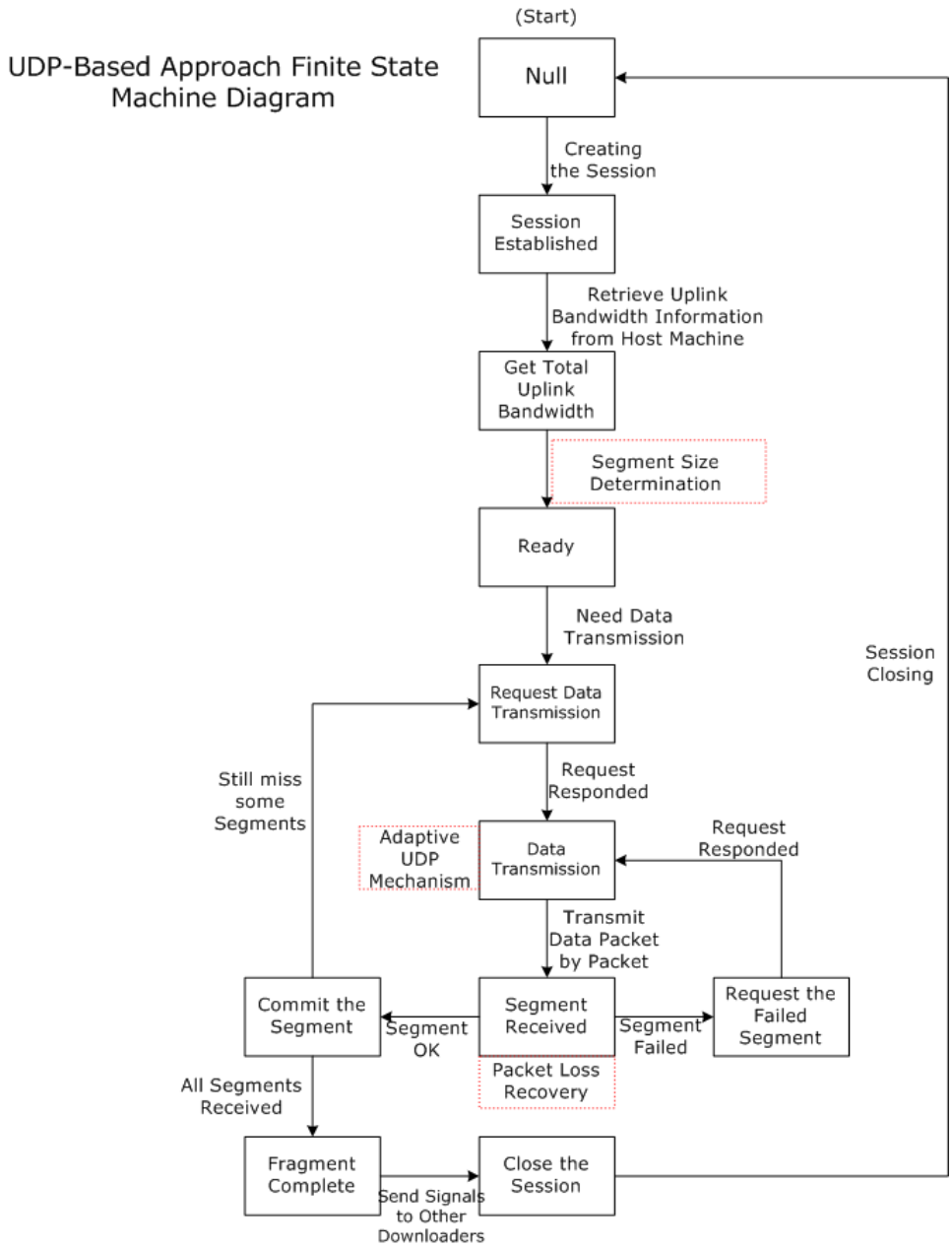


圖 3. 1：UDP-Based Approach FSM Diagram

我們設計的新協定中，想要提高之效能目標為：

(1) 儘量降低重傳次數。

(2) 儘量利用可利用之網路頻寬，提高每個下載者的下載速度。

為了追求我們所設定之效能目標，在所設計的協定中，包含了三項與效能有關之運作機制，其分項說明如下：

(1)Packet Loss Recovery (自動重建遺失之封包)： 為避免封包遺失引發太多的 Fragment 重傳，我們加入自動恢復機制。

(2)Segment Size Determination (傳送單位估算)：計算檔案片段(Fragment)中，最佳的基本傳輸單位(Segment)大小，使得總 Overhead 為最低。

(3)Adaptive UDP Mechanism (調節式 UDP 機制)：彌補 UDP 沒有自動調整傳輸速率的問題。我們希望能根據網路狀態，將資料傳送速度調整到最適的傳輸速率。

3.1 Packet Loss Recovery

如圖 3.2，檔案片段結構如下：

(1)一個檔案片段，總共有 m 個封包 (未加上同位封包時)。

(2)一個 Segment 內含 n+1 個 封包。一個檔案片段是由一個以上的 Segment 組合而成。

如下圖所示：

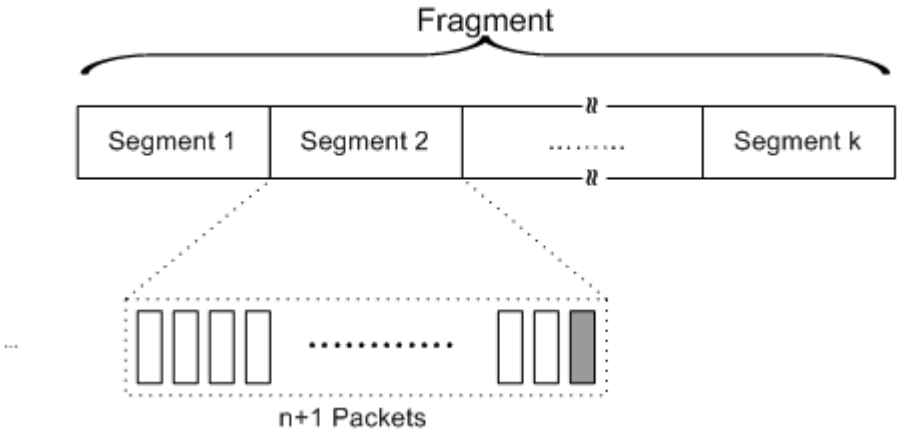


圖 3. 2：檔案片段結構

為了避免封包損傷導致太多的資料重傳，我們設計了一個自動重建遺失封包之機制(Packet Loss Recovery)。在每 n 個封包之後，加上一個同位封包 (Parity Packet)，同位封包上的每一個 bit 是由 n 個封包中相對應的 bit 用同位計算的方式所產生。這 n+1 個封包為一個錯誤更正的單位，稱為一個 Segment。而當一個 Segment 中任何一個封包發生遺失時，就可利用同位封包將所遺失的封包還原。

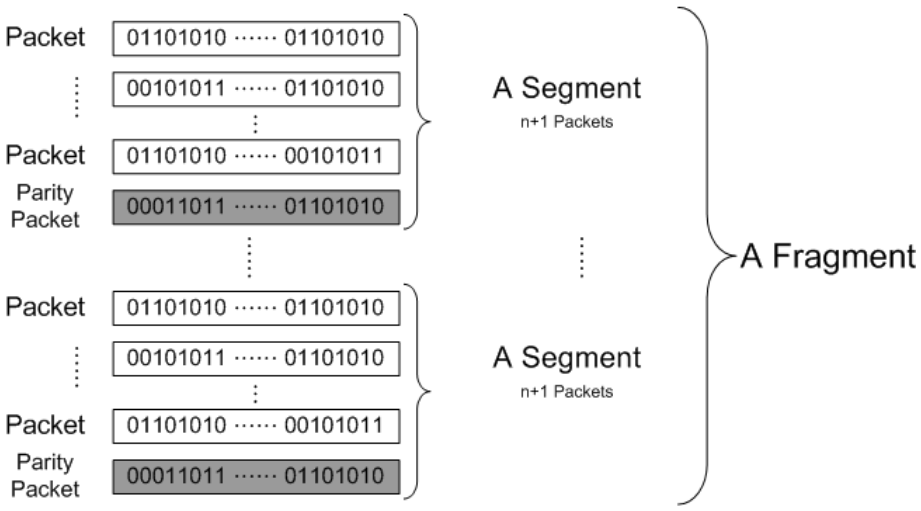


圖 3. 3：Packet Loss Recovery

當傳輸一個檔案片段時，任一 Segment 中若有超過一個封包出現錯誤，同位封包將無法彌補，此 Segment 必須重傳。Segment 之重傳由接收端對無法利用同位封包重建的檔案片段，發出重新傳送之請求，要求傳送端再發送一次該檔案片段。其細部運作流程如下圖 3.3 所示：

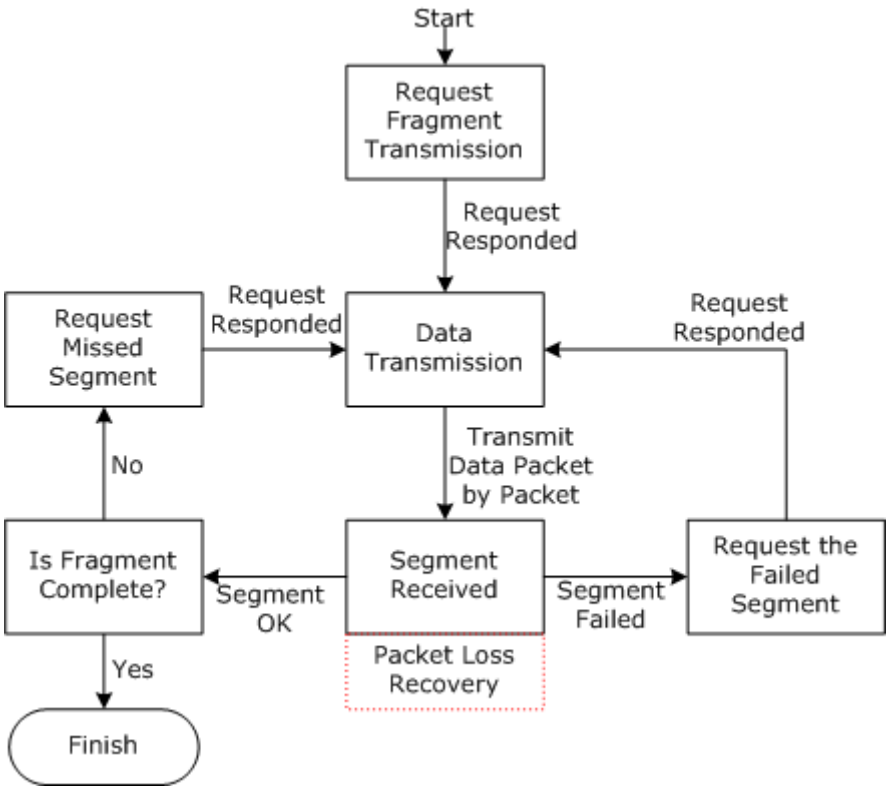


圖 3. 4：Packet Loss Recovery 運作流程圖

本協定之設計是架構在 UDP 協定與應用程式之間(如同 RTP 協定[18][20]一般)。P2P 應用系統將其資料送至本協定時，本協定將之切成封包大小一樣之封包，並逐一加以編號再送由 UDP 協定傳送資料。接收端傳輸協定收到封包後，依同位封包檢查得到的結果，決定 Segment 是否需要重傳。若資料正確，則將其收到資料送至 P2P 應用系統進行後序的動作。若不正確，接收端對此 Segment

提出重送之請求。

Segment 中之 Segment Size 對協定之運作有影響，若 Segment Size 較小時，錯誤更正能力較強，但 Overhead 變大。若 Segment Size 較大時， Overhead 較小，但錯誤更正能力也相對較弱。最佳 Segment Size 之決定將在下節中討論。

3.2 Segment Size Determination

上節曾提及 Segment Size 大小會影響傳輸效率。本節介紹我們所設計尋找最佳 Segment Size 之計算法，所定義之參數如表一。

表 3.1： 最佳 Segment Size 計算法符號

符號	意　　義
m	一檔案片段(Fragment)中之封包數
γ	封包遺失率，$0 \le \gamma \le 1$
n	一個 Segment 之封包數

我們推導一個計算公式。以 BitTorrent-like 檔案共享系統之最佳傳輸效率為目標，檔案片段之封包數、封包遺失率為已知參數。

(1) 一個 Segment 中， x 個封包遺失的機率

一個 Segment 的 $n+1$ 個封包中有 x 個封包遺失的機率可以寫成下列的二項式分配(Distribution)：

$$bin(x \mid n+1, \gamma) = \binom{n+1}{x} \gamma^{x} (1-\gamma)^{(n+1-x)} \qquad (1)$$

(2) 一個 Segment 傳送成功的機率

因為我們設計錯誤更正機制的單位為 Segment。一個 Segment，可救回一個遺失或錯誤的封包。所以，一個 Segment 的成功機率為「沒有封包遺失的機率加上只有一個封包遺失的機率」，如下：

$$bin(0 \mid n+1, \gamma) + bin(1 \mid n+1, \gamma) \qquad (2)$$

(3) 反之一個 Segment 傳送失敗的機率為：

$$p = 1 - \left(bin(0 \mid n+1, \gamma) + bin(1 \mid n+1, \gamma) \right) \qquad (3)$$

(4) 額外成本

一個 Fragment 可分為 $\left\lceil \frac{m}{n} \right\rceil$ Segment。一個 Segment 中需增加一個同位封包，成本為 1。當一個 Segment 傳送失敗時，仍要再重傳一次，其重傳成本為 $n+1$。

我們可設計一個懲罰函數(Penalty Function)：

$$Penalty(n \mid m, \gamma) = \left\lceil \frac{m}{n} \right\rceil \left[1 + (n+1)p + 2(n+1)p^{2} + 3(n+1)p^{3} + \cdots \right]$$

$$, \ 1 \le n \le m . \qquad (4)$$

整理化簡：

$$\Rightarrow Penalty(n \mid m, \gamma) = \left\lceil \frac{m}{n} \right\rceil \left[1 + \frac{(n+1)p}{(1-p)^2} \right] \quad (5)$$

(5)當 $Penalty(n \mid m, \gamma)$ 最小時，可得佳 n 值，故我們利用微分求極小值。

$$\frac{d}{dn} Penalty(n \mid m, \gamma) = \frac{d}{dn} m \left[\frac{1}{n} + \frac{(n+1)p}{n(1-p)^2} \right] = 0 \quad (6)$$

$$\Rightarrow \frac{d}{dn} Penalty(n \mid \gamma) = \frac{d}{dn} \left[\frac{1}{n} + \frac{(n+1)p}{n(1-p)^2} \right] = 0 \quad (7)$$

給定一 γ 值代入上式，求解後再對 n 值四捨五入即可解得最佳 Segment Size 值。

3.3 Adaptive UDP Mechanism

由於 UDP 協定自身沒有決定傳送速率的機制，我們必須在此協定中加入自動調整速率之機制，為此，我們提出 Adaptive UDP 機制。在實際環境中，影響資料傳送速率的一個最重要因素，即是瓶頸頻寬(bottleneck bandwidth)。由於本文主要是針對 P2P 檔案共享系統在非對稱網路環境之下的效能改進，路徑中瓶頸的發生點有兩個：(1)發生在使用者上行端，或者(2)發生在網路內部。為處理瓶頸發生在網路內部之情況，我們需利用探測封包(Probing Packet)的方式，來幫助我們瞭解網路內部瓶頸頻寬(bottleneck bandwidth)的狀況。

3.3.1 與頻寬相關之 Metrics

首先，我們先釐清兩個名詞之間的差異[7]：

(1)頻寬(Capacity)：鏈結或路徑上之最大資料傳送能力。

(2)可用頻寬(Available Bandwidth)：鏈結或路徑上未被使用之頻寬。

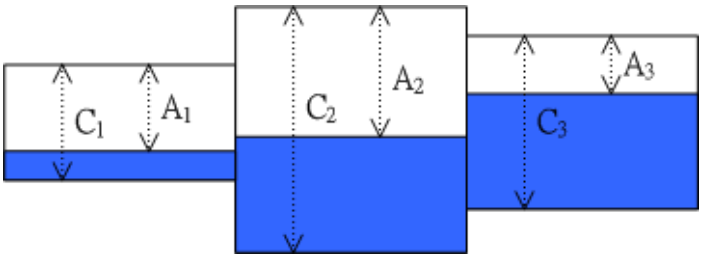


圖 3. 5：UDP with Probing Packet

如圖 3.4 所示，C_1、C_2、C_3 是每個鏈結上之頻寬，而 A_1、A_2、A_3 是每個鏈結上之可用頻寬。而我們所要量測之瓶頸頻寬(Bottleneck Bandwidth)，為可用頻寬中最窄者，也就是圖中之 A_3。

我們 Adaptive UDP Mechanism 的設計，傳送端需要定期發送探測封包量測網路狀態，估計合理的傳送速率，接收端在 ACK 回傳時告知所測得之速度。傳送端再使用此估計出來的最適速率傳送資料，可以降低頻寬浪費或網路擁塞之機率。

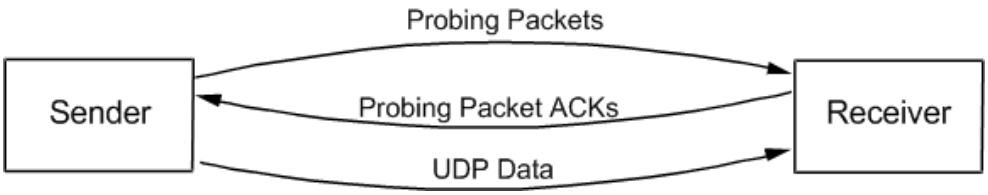


圖 3. 6：UDP with Probing Packet

Constantinos Dovrolis 等人在"Packet-Dispersion Techniques and a Capacity-Estimation Methodology." [3]中提到，緊鄰送出的兩個封包，在通過瓶頸

鏈結時，會因為瓶頸鏈結處理限制以及佇列中其他封包之影響，其封包距離因而有散開(Dispersion)的現象，此散開的值可當做目前瓶頸可用頻寬的估計依據，此法稱為 Packet Dispersion 法。我們利用此 Packet Dispersion 的方式估計目前網路內部瓶頸的頻寬(Estimated_Bandwidth)，然後和非對稱網路中頻寬較窄的上行頻寬相比較之後，取其值較小者，亦即：

$$Available\_Bandwidth = \min(Uplink\_Bandwidth, Estimated\_Bandwidth) \quad (6)$$

由上式計算所得之值，來當做我們實際可用的頻寬。Adaptive UDP 即是利用此 Available_Bandwidth 所提供的資料為指標，進行資料傳送速率(Data Rate)的調整。

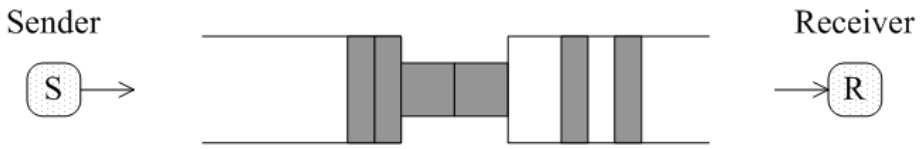


圖 3. 7： Packet Dispersion

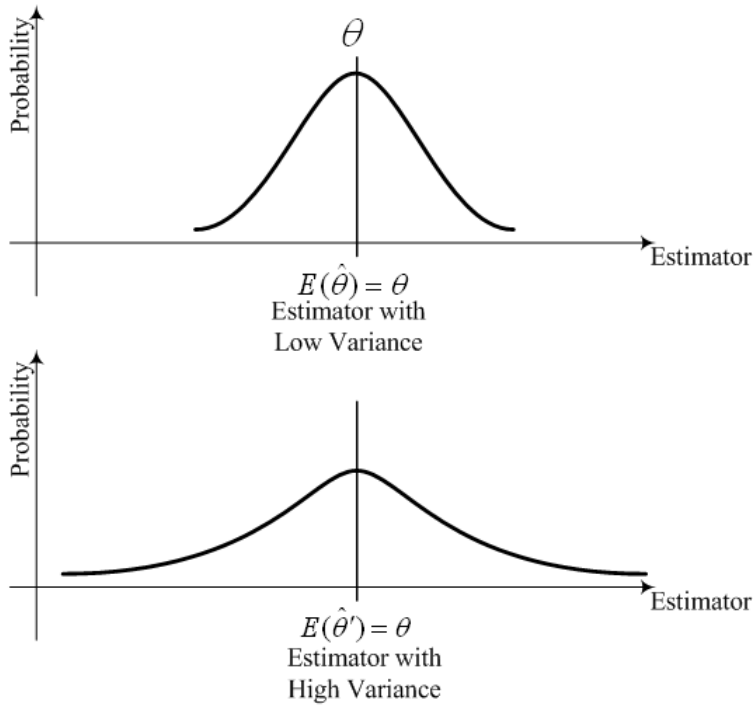


圖 3. 8： Variation Notation

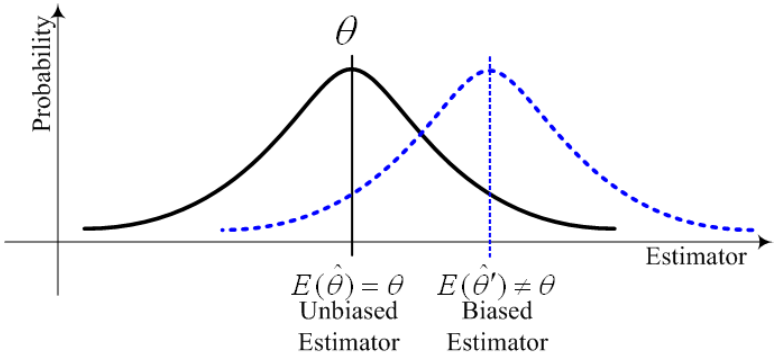


圖 3. 9： Bias Notation

此處，我們為了避免估計偏差性(bias)對我們協定造成負面影響，使用一個修正係數 α (Network State Adjusting Coefficient α)用以校正估計結果。因為 Packet Dispersion 是經由使用者端用 End-to-End 的方式去估計網路頻寬，由於沒有透過網路設備(如 Router Assistant)，所收集到的訊息較不充分。為了修正估計誤差，在利用 Packet Dispersion 法估計出來的頻寬時，須再乘上一修正係數 α 。

此α係數的值，即是為了反應網路狀況的一個校正參數。而且，我們接收端量測到的 Packet Dispersion 的值，可以利用它的 Dispersion Time，用以估計目前瓶頸剩餘的可用頻寬(Available Bandwidth)，之後再利用修正係數 (Adjusting Coefficient) α來校正其估計值。如下：

$$Estimated\_Bandwidth(bytes/sec) = \alpha \times \frac{Packet\_Size(bytes)}{Average\_Dispersion\_Time(sec)}$$

最後比較上行頻寬(Uplink Bandwidth)和所測得的估計頻寬(Estimated Bandwidth)，取其較小值，做為傳送速率。UDP 協定內並無類似 TCP 內 congestion window size 的機制以控制傳送速度。我們把傳送速度之控制建在應用層中。

此外，校正參數α所訂定出來的數值對整個資料傳輸行為有重要的影響，故在訂定校正參數α值時需小心謹慎。另外，我們所測得之可用頻寬為調整資料傳送速度之一參考指標，其調整方式，可參考[12]。

第四章

效能評估

本研究所設計的網路協定中， Segment Size Determination (傳送單位之計算)需要進行參數之計算， Adaptive UDP Mechanism(調節式 UDP 機制)需要進行模擬實驗估算其校正參數。最後，再以第三章所提之 UDP-Based Approach 作為實驗組，以 TCP-Reno、TCP-Vegas 作為對照組，在不同的訊務量及不同的網路封包遺失率之下，進行效能評估，並根據實驗結果進行評論。

4.1 模擬與計算環境

我們實驗所採用的模擬環境，是 Cygwin 下的 NS (Network Simulator) 2.28 版，並且使用 Perl 5.8.7 作為剖析實驗記錄檔的工具，最後觀察實驗剖析得到之結果並評論之。

4.2 參數估算

4.2.1 傳送單位之 Segment Size 計算

實驗目標：為提昇協定之下載速率，我們給定特定的網路環境，利用最小化懲罰函數找出最佳 Segment Size。以及觀察在不同的網路封包遺失率之下，對 Segment Size 變化影響之敏感度分析。

4.2.1.1 Segment Size 計算實驗設計

我們利用數學軟體 Matlab，根據傳送單位之計算計算機制的推導結果，設計了一個懲罰函數 (Penalty Function) 程式，計算最佳 Segment Size。由於 BitTorrent 設定檔案片段大小為 1/4 Mb，故我們計算範例所設定的參數為：

表 4. 1 :Segment Size 計算之實驗參數

參數	數　值　範　圍
m	263 packets/fragment
γ	0.001 ~ 0.015
l	1000 bytes

4.2.1.2 Segment Size 計算結果

經過 Matlab 計算後，可得到下面的關係圖：

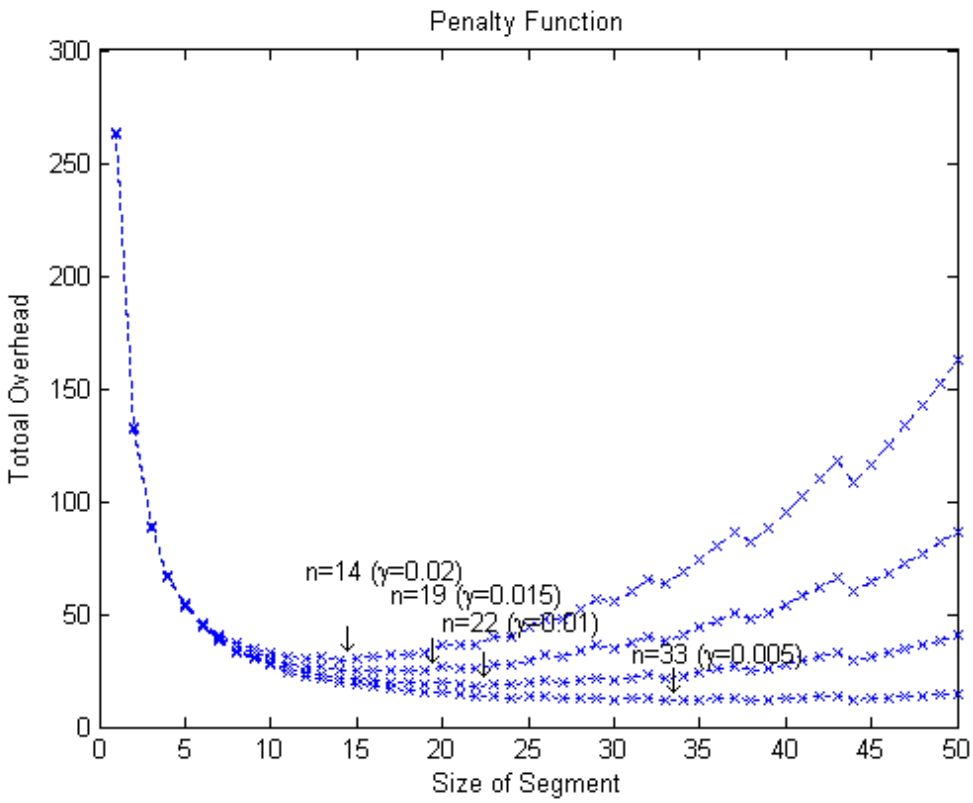


圖 4. 1：Segment Size 對 Total Overhead 之影響 (γ =0.001, 0.005, 0.001, 0.015)

其中，橫軸為 Segment Size 的變動，縱軸為懲罰值(Penalty Value)。如圖 4.2 中，在此範例之 γ 值若為 0.005，Segment Size 為 33 時，有最小的 Total Overhead。

4.2.1.3 Segment Size 計算之敏感度分析

接下來，我們欲研究網路於不同的網路封包遺失率之下，對 Segment Size 變化影響之敏感度分析。故我們將封包遺失率值設定為 0.001～0.015 變化，與最佳 Segment Size 之變化情形繪圖，可得到如下的結果：

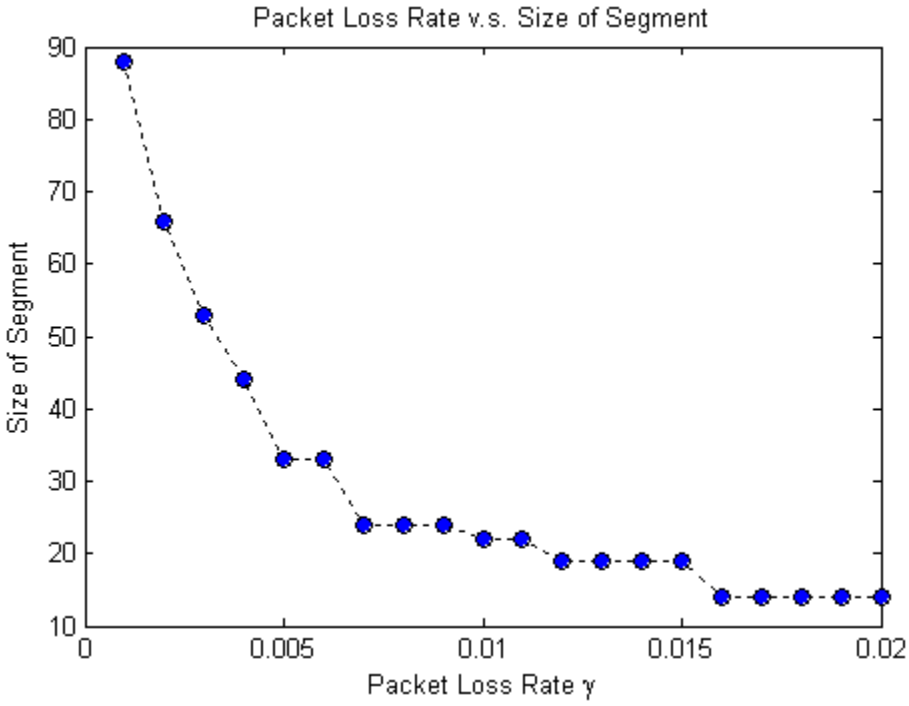


圖 4. 2：封包遺失率對 Segment Size 之影響

由上圖可以看到，封包遺失率(Packet Loss Rate)很低的情況下，不太會發生封包遺失，所以求得的 Segment Size 比較大，表示此時資料傳輸時，並不需要太多的同位封包(Parity Packet)來保護資料封包。但是，如果封包遺失率提高時，因為封包遺失就容易發生，求得之 Segment Size 就變小了，表示此時需要較高比率之同位封包來保護資料封包。所以，上圖所繪 Segment Size 對應封包遺失率的趨勢圖，符合原先我們所預期的結果。

4.2.2 調節式 UDP 機制之校正參數估算

實驗目標：設計一個魚骨狀之網路拓樸，利用 Packet-Dispersion 之方式估計可用頻寬，並且經由實驗得到經驗之校正參數 α 值供使用者參考。

4.2.2.1 校正參數 α 值估算實驗設計

實驗的場景設定若瓶頸鏈結發生在核心網路(Core Network)時，利用 Packet Dispersion 的方式去估計可用頻寬所得到之值會有誤差。在我們的設計中，利用 α 來修正此誤差。我們所設計之網路拓樸為一魚骨狀之網路架構。中間的部份為中介路由器，中介鏈結上有中介訊務流經其中模擬漂經路由器之其他訊務流。我們調整此魚骨拓樸之長度，並且控制實際可用頻寬之下，求取參考用之校正參數 α 值。

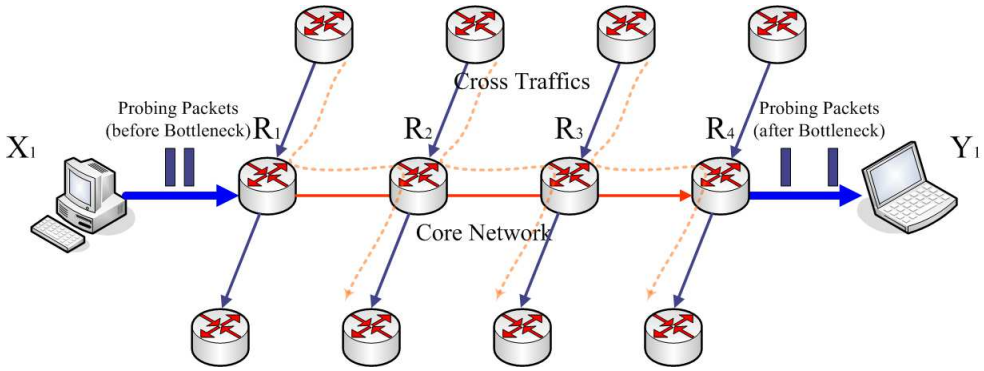
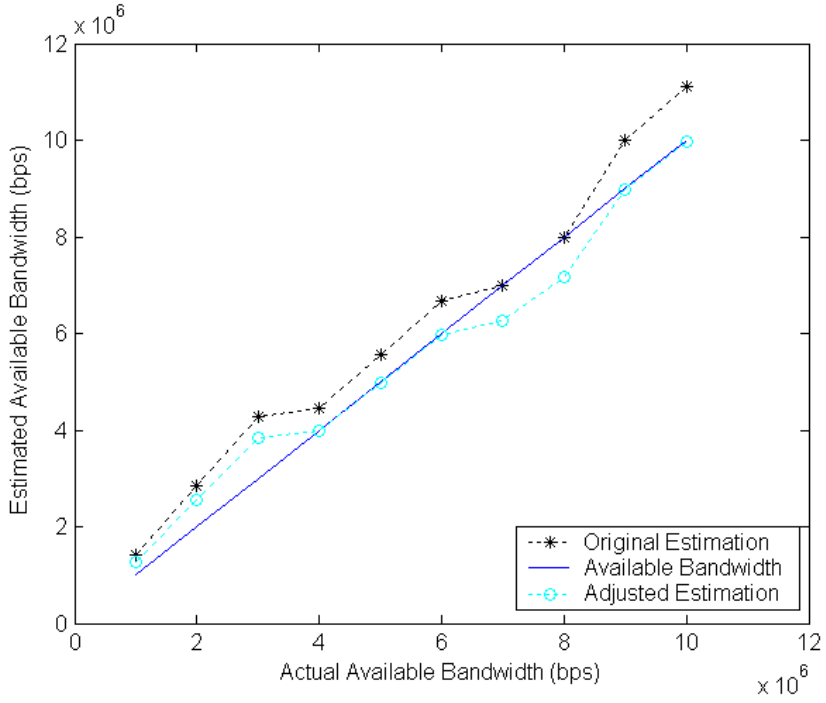


圖 4. 3：Adjusting Coefficient α Simulation Topology

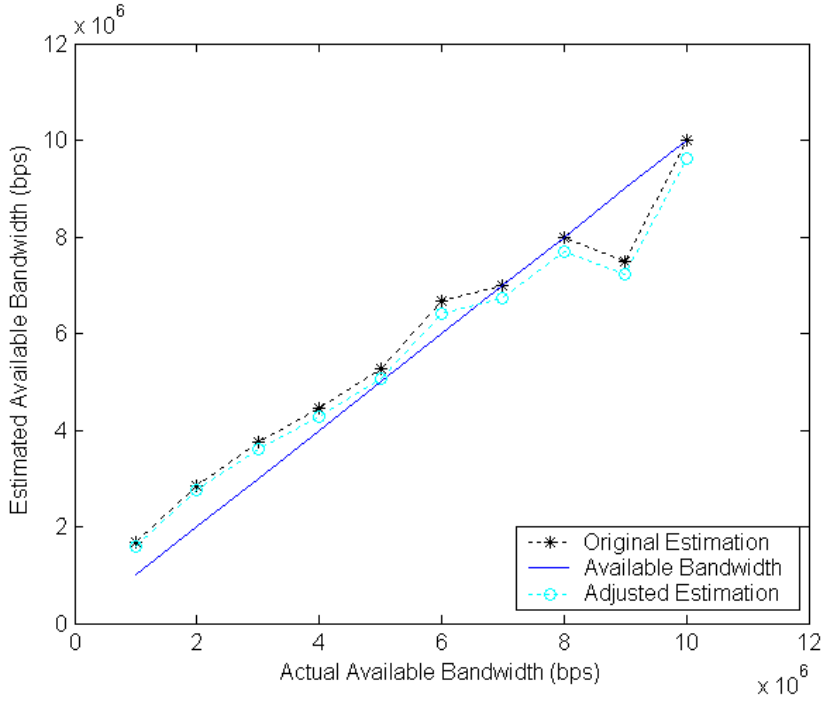
表 4. 2 :估計校正參數 α 值之實驗參數設定

參數	數　值　範　圍
#L	1,3,5,7,9
A	1～10 Mbps。

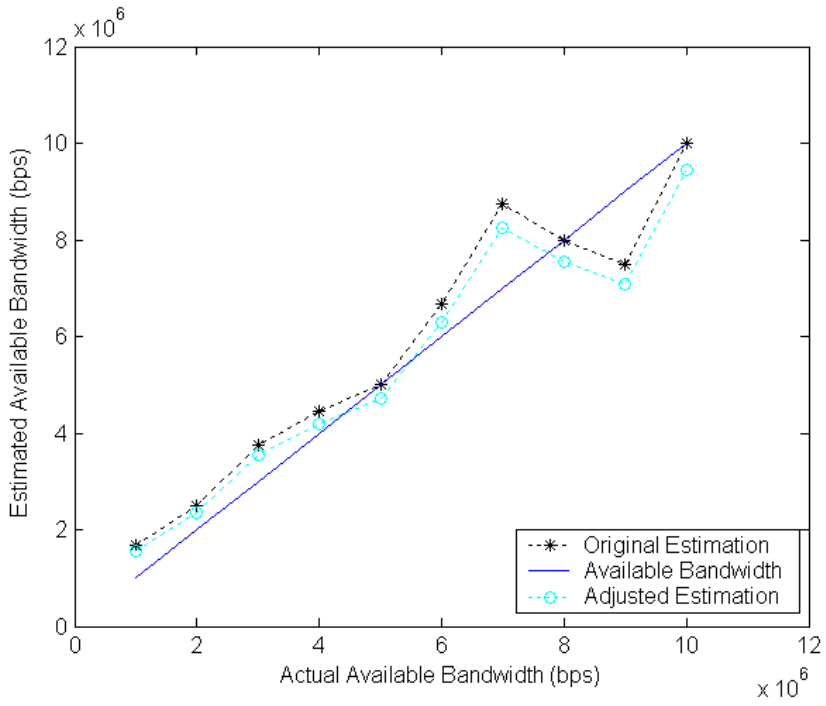
4.2.2.2 校正參數 α 值估算實驗結果



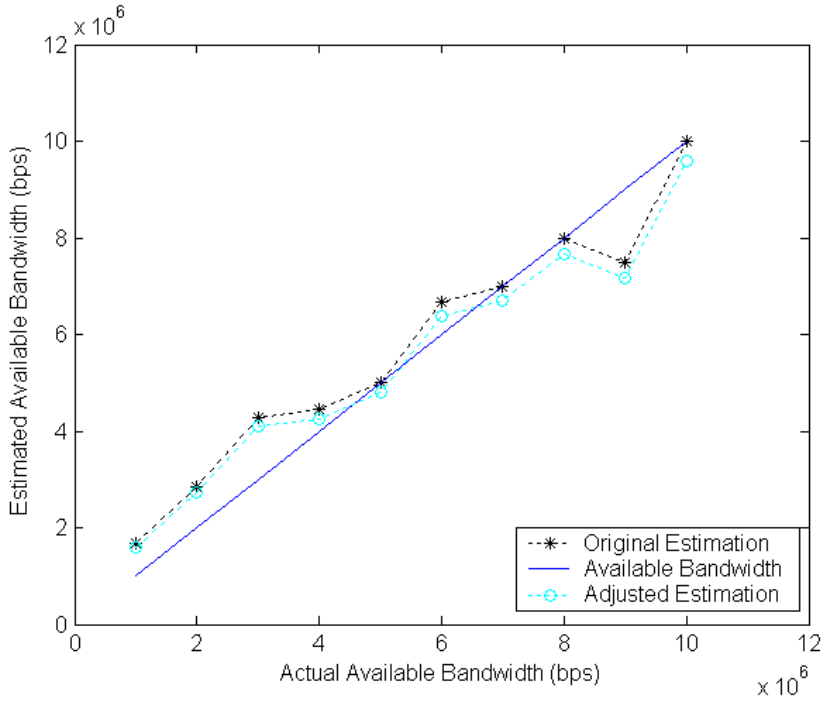
(a) 中介鏈結數=1



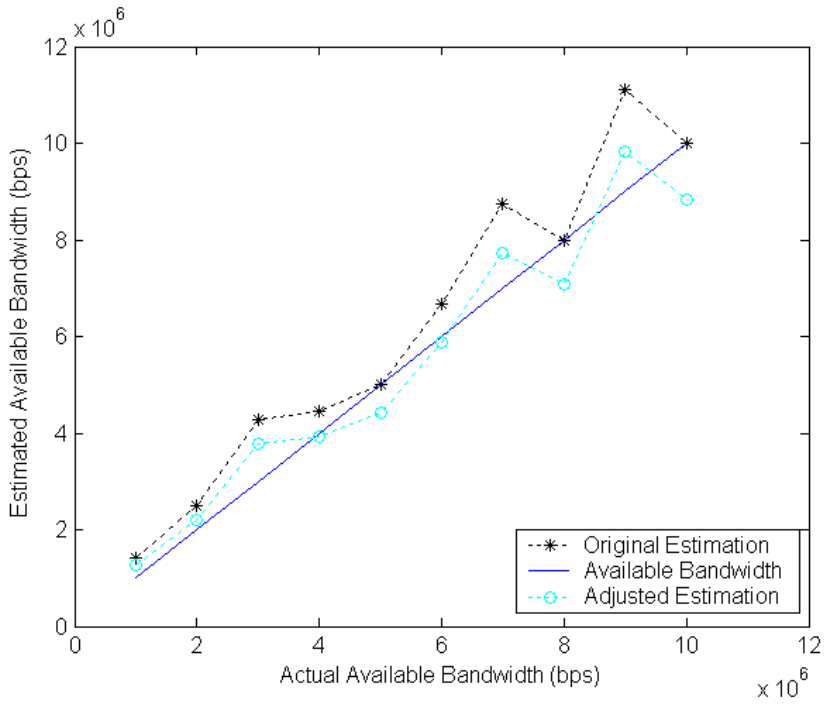
(b) 中介鏈結數=3



(c) 中介鏈結數=5



(d) 中介鏈結數=7



(e)中介鏈結數=9

圖 4. 4：校正參數 α 值估算之實驗結果(a)中介鏈結數=1, (b)中介鏈結數=3, (c)中介鏈結數=5, (d)中介鏈結數=7, (e)中介鏈結數=9.

本實驗是利用 Packet-Dispersion 法估計可用頻寬，由圖 4.5 可看到此法所估計到之頻寬皆略有誤差。經由我們所設計利 A 與 $\hat{A}$ 間之關係求得之校正參數 α 校正之後可避免此項誤差，可看到圖中校正過之估計值(Adjusted Estimation)比原始估計值(Original Estimation)更接近實際可用頻寬(Actual Available Bandwidth)。

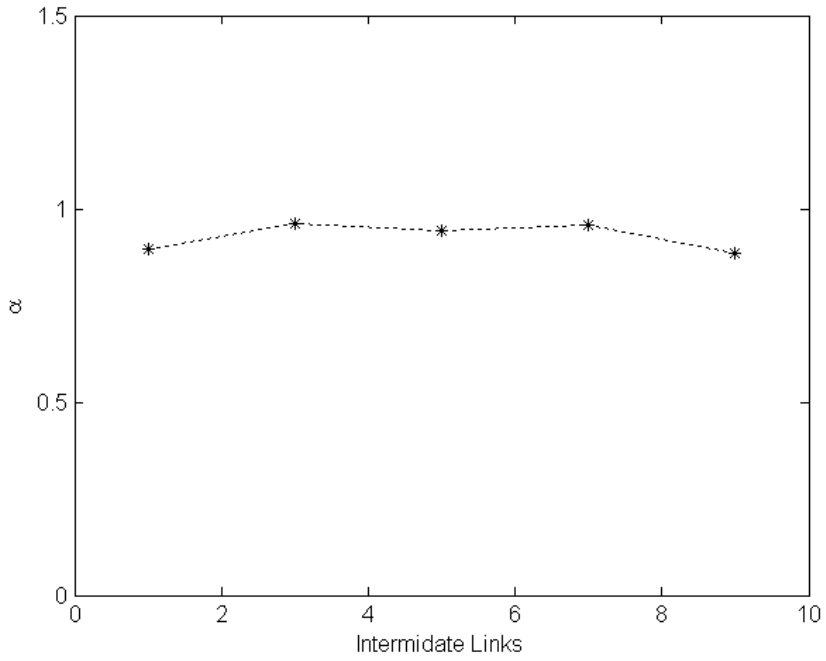


圖 4. 5：中介鏈結數之變化與校正參數α之間的關係

我們觀察中介鏈結數變化與校正參數α之間的關係，如圖 4.6，可發現α值相當平穩，由此推測中介鏈結數之大小對α值的影響不大。最後實驗過程中得到最後估計之α值為 0.929。

4.3 效能評估

以 UDP-Based Approach 作為實驗組，以 TCP Reno、TCP Vegas 作為對照組，在不同的訊務量及不同的網路封包遺失率之下，進行效能評估，並根據實驗結果進行評論。我們設計一個網路上可能會發生 FUB 與 BoA 的非對稱網路 P2P 檔案分享系統之場景，並且對 UDP-Based Approach、TCP Reno、TCP Vegas 進行效能分析與比較。

4.3.1 評估指標

量測每個參與者下載完成一個檔案片段之時間。若平均完成時間短，表示有較佳之效能表現；平均完成時間長，則為較差之效能表現。

4.3.2 效能評估實驗設計

實驗的場景設定在核心網路(Core Network)中有充沛頻寬，雙向有 1 Gbps 頻寬，而非對稱網路則為下載 (2~8) Mbps、下行 256 Kbps 的頻寬。在檔案分享系統中，每個檔案片段為 1/4Mbps。為了研究前述文中所提出在不同網路環境下的 FUB 與 BoA 對效能影響的問題，我們設計了一個非對稱網路下檔案分享系統的網路拓樸。

如下圖，核心網路內部設定了不同之封包遺失率之變化，而核心網路有 R1、R2 兩個路由器(Router)分別接至外部非對稱的鏈結上，非對稱鏈結所接取的的機器有 X_1～X_5及 Y_1共六台，其中 X_1～X_5共有五個 Session 會傳送資料至 Y_1 (Session 1~5)。而 Y_1 亦有五個 Session 分別傳送資料至 X_1～X_5 (Session 6~10)，如此在 Y_1 至 R2 的上行鏈結有可能會發生 FUB 與 BoA 的問題，我們即可量測在 UDP-Based Approach、TCP Reno、TCP Vegas 等傳輸協定之下，P2P 檔案分享傳輸應用系統效能之分析比較。

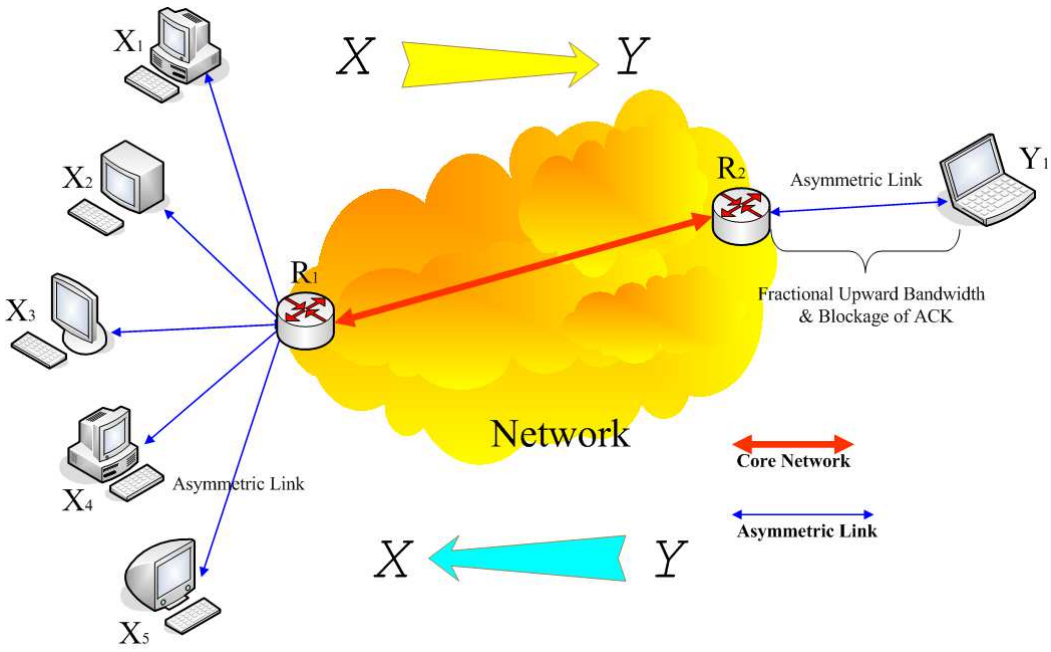


圖 4. 6：Performance Evaluation Simulation Topology

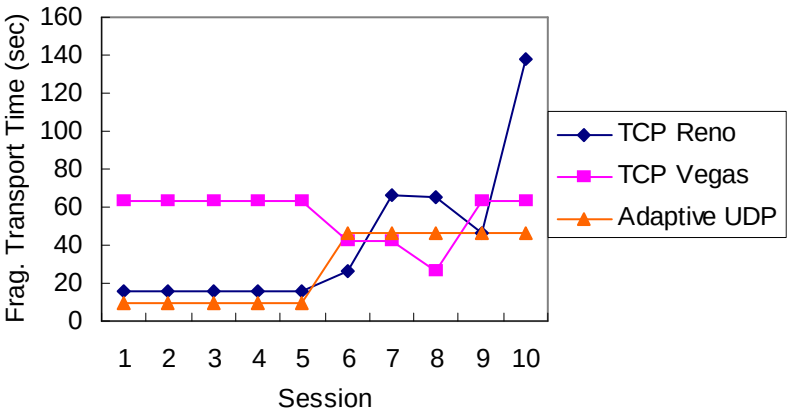
其參數設計如下：

表 4. 3 :效能評估之實驗參數設定

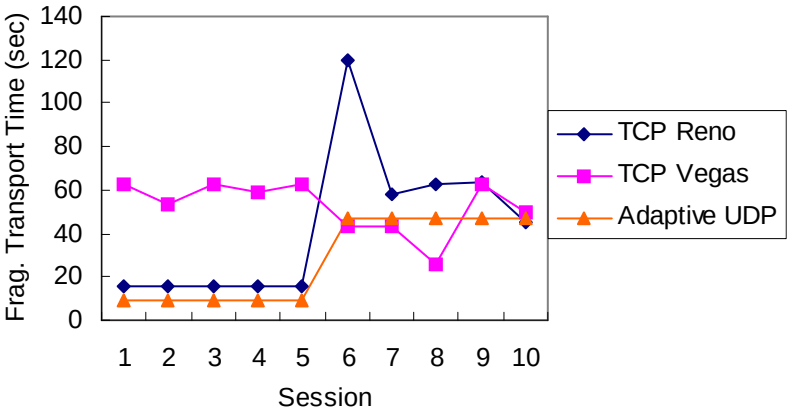
參　　數	數　　值
傳輸協定	TCP-Reno、TCP-Vegas、UDP-Based Approach
核心網路頻寬	1 Gbps
接取網路上行下行頻寬	(256 Kbps, 2Mbps)、(256 Kbps, 4 Mbps)、(256 Kbps, 6Mbps)、(256 Kbps, 8 Mbps)
封包遺失率	0%, 0.1%, 0.5%, 1%, 5% 10%

4.3.3 效能評估實驗結果

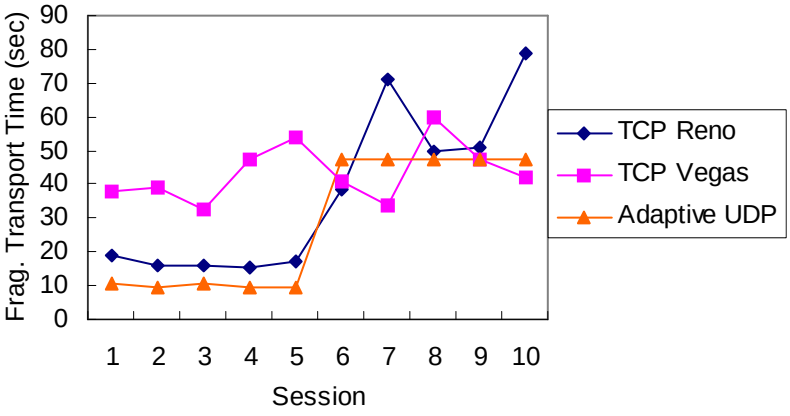
在經過用 ns2 對各種傳輸協定進行模擬之後，比較不同協定中各個 Session 中下載一個檔案片段 1/4MB 資料所需花的時間。



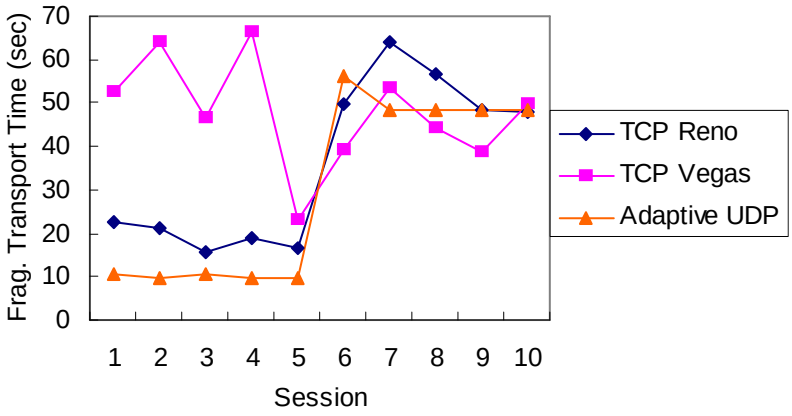
(a) γ =0



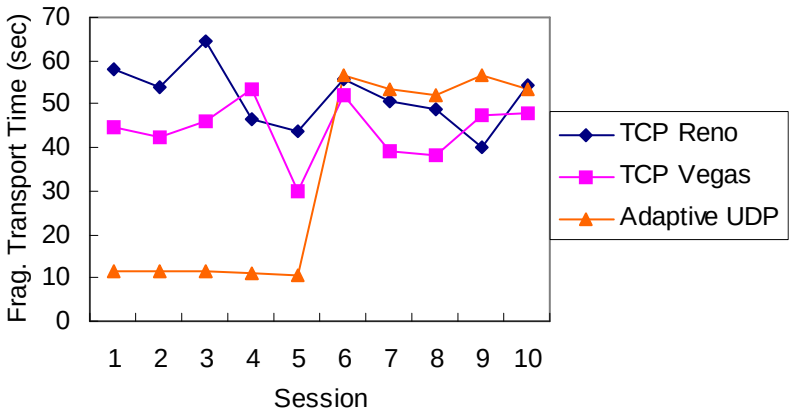
(b) γ =0.001



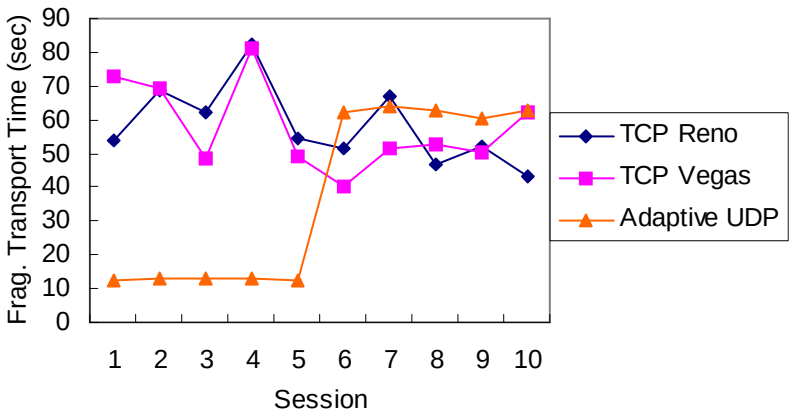
(c) γ =0.005



(d) γ =0.01

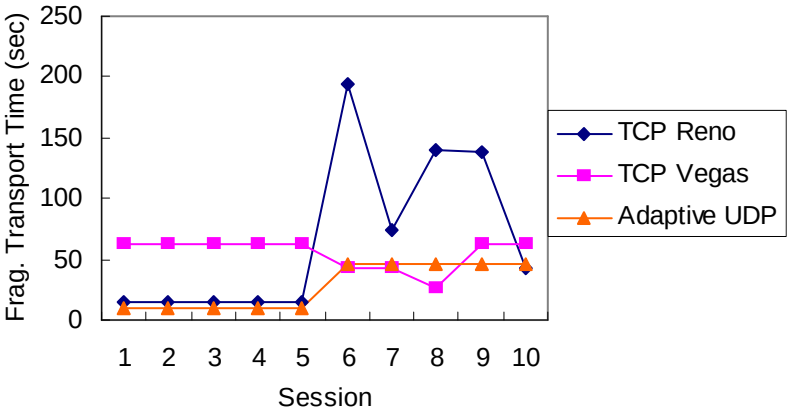


(e) γ =0.05

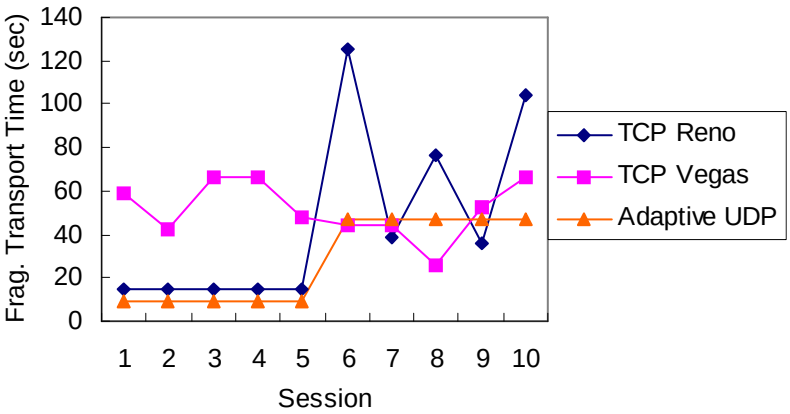


(f) γ =0.1

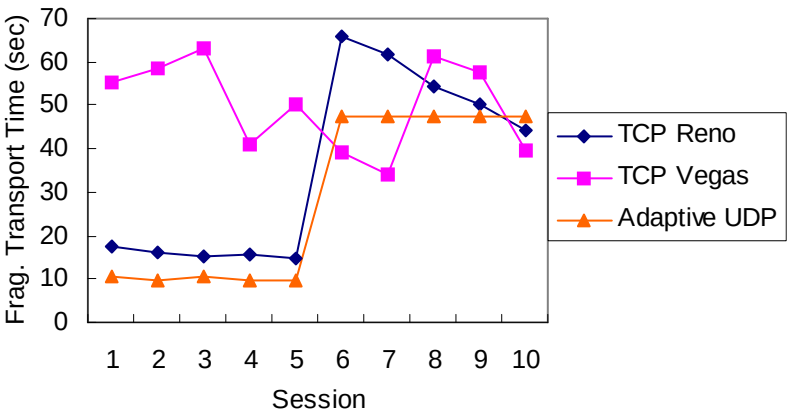
圖 4. 7：上行 256Kbps,下行 2Mbps, γ 變動下不同傳輸協定效能之比較



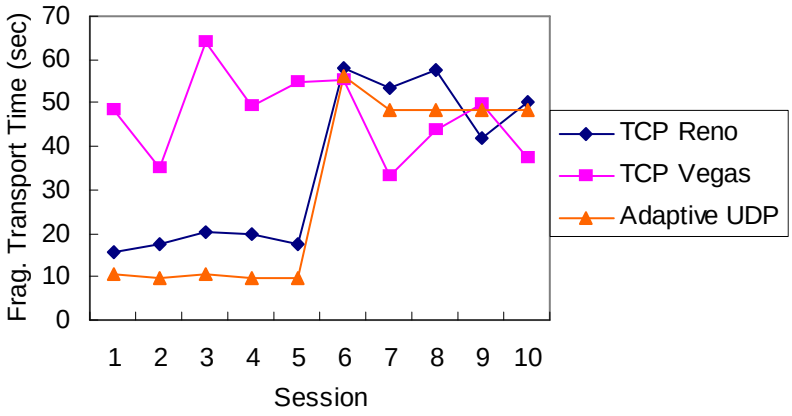
(a) γ =0



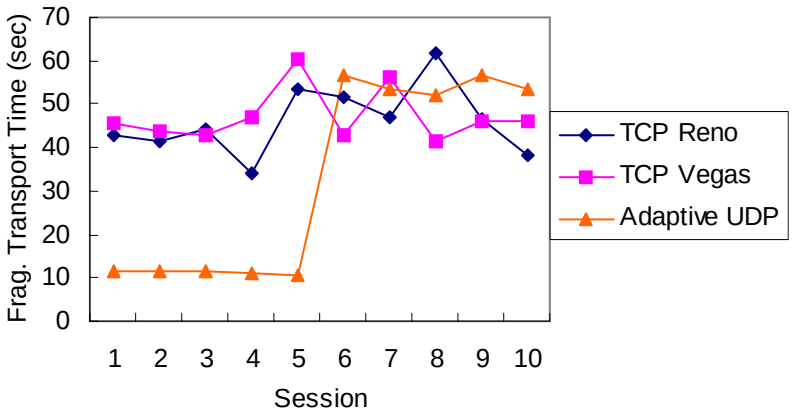
(b) γ =0.001



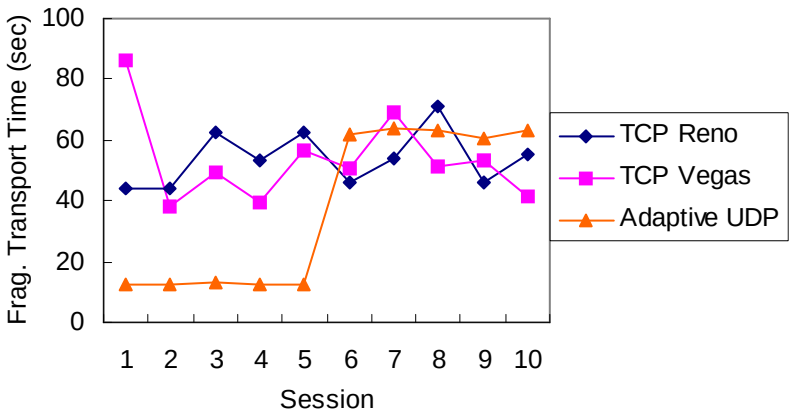
(c) γ =0.005



(d) γ =0.01

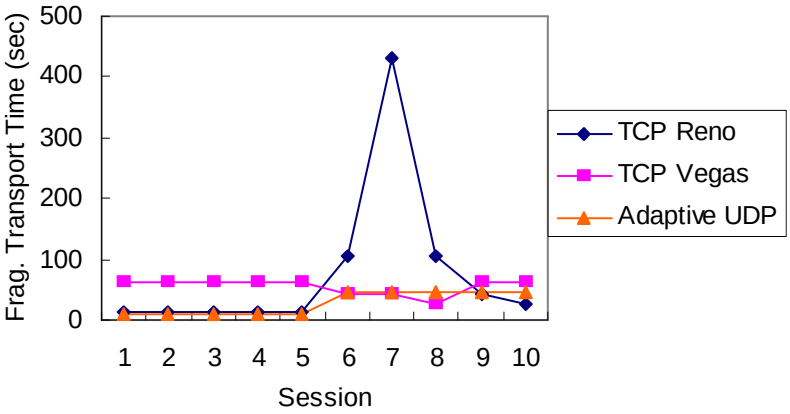


(e) γ =0.05

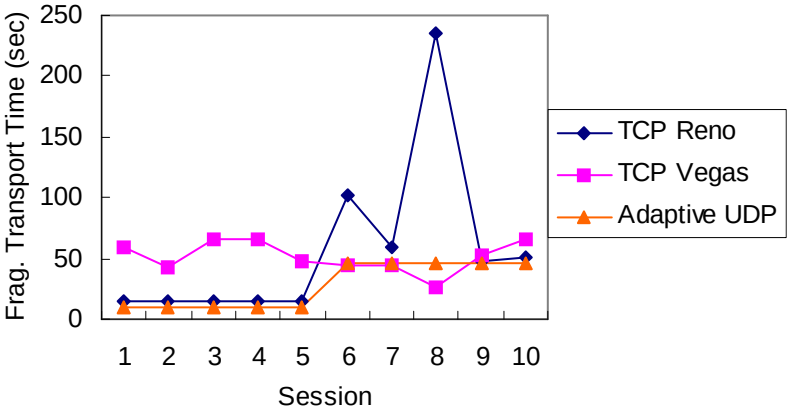


(f) γ =0.1

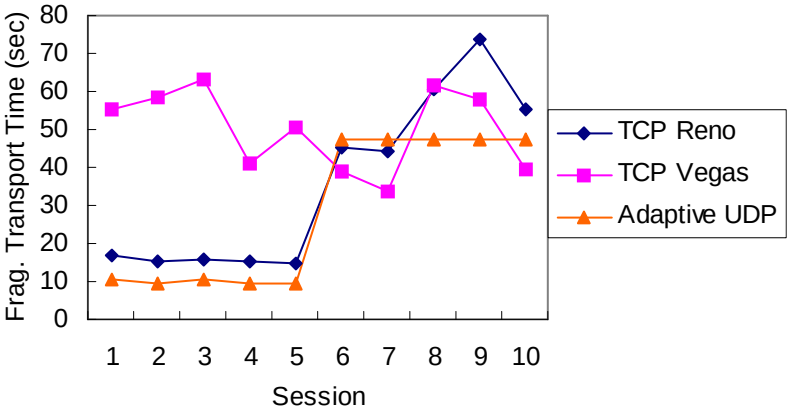
圖 4. 8：上行 256Kbps,下行 4Mbps, γ 變動下不同傳輸協定效能之比較



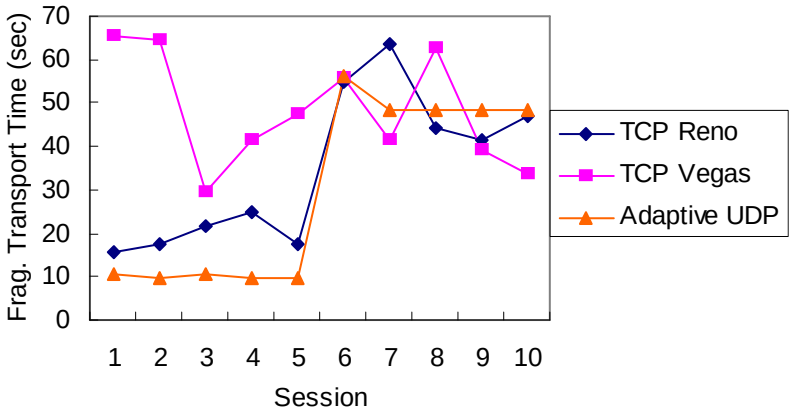
(a) γ =0



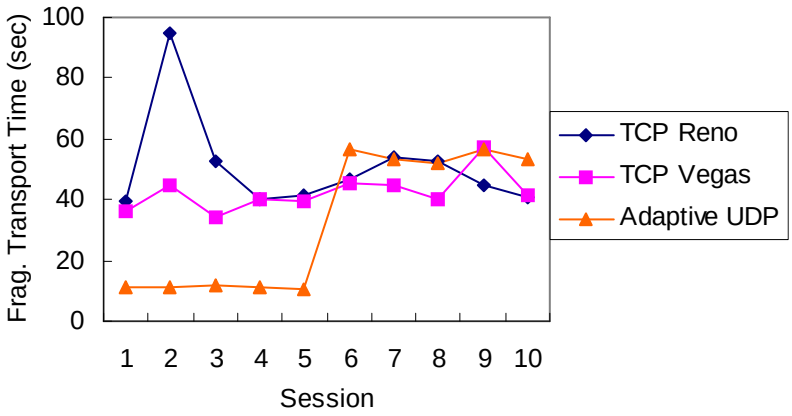
(b) γ =0.001



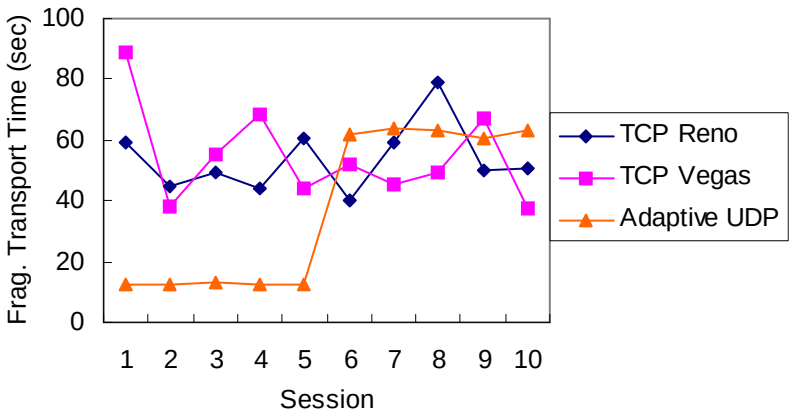
(c) γ =0.005



(d) γ =0.01

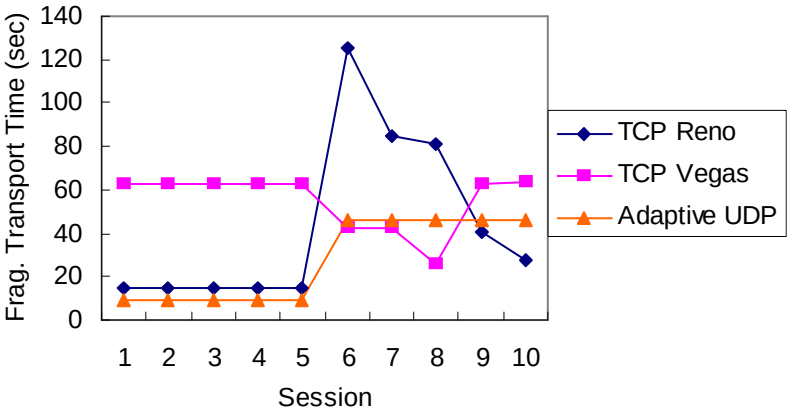


(e) γ =0.05

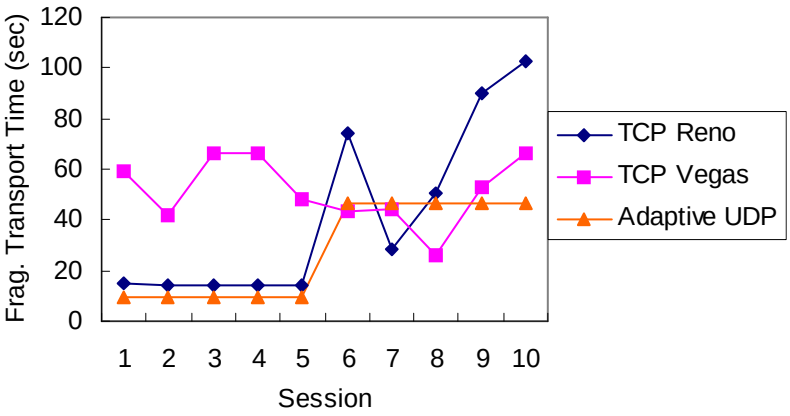


(f) γ =0.1

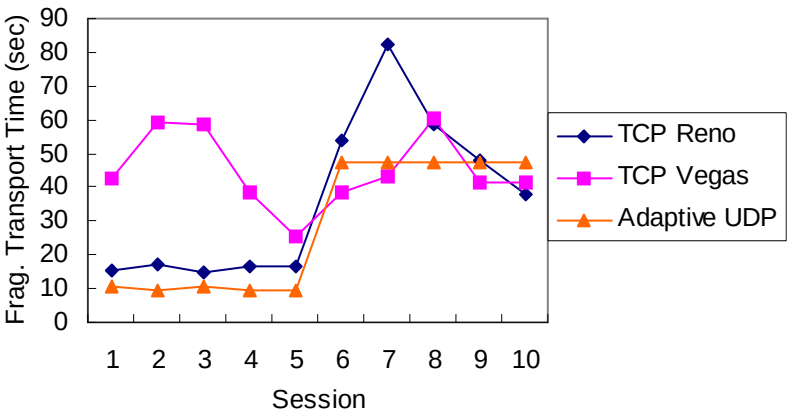
圖 4. 9：上行 256Kbps,下行 6Mbps, γ 變動下不同傳輸協定效能之比較



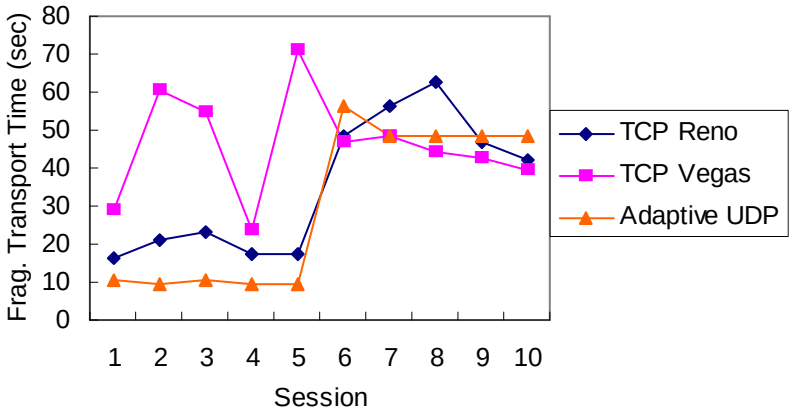
(a) γ =0



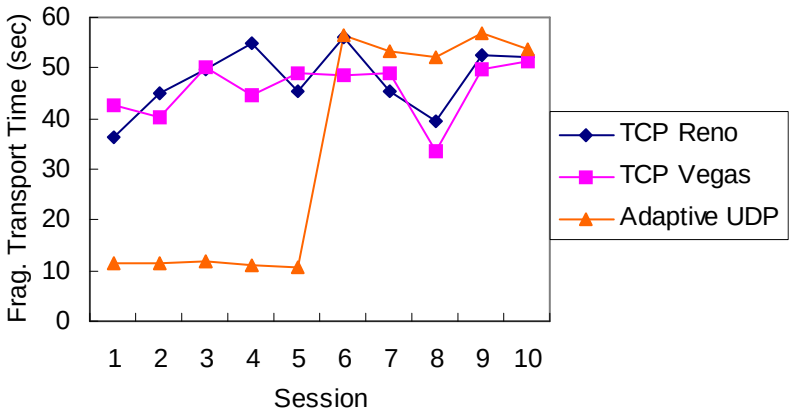
(b) γ =0.001



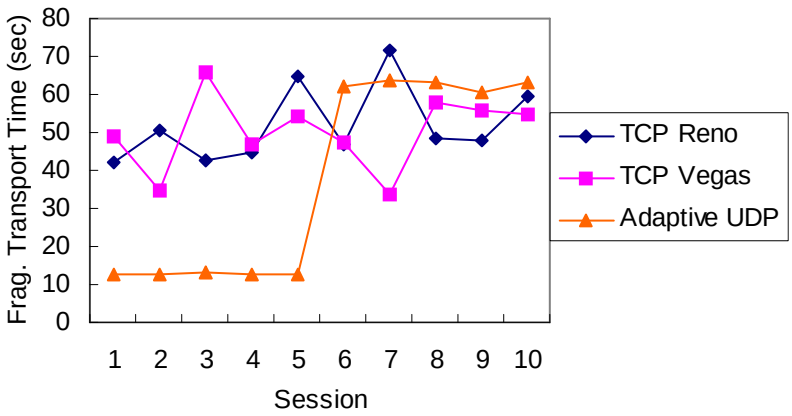
(c) γ =0.005



(d) γ =0.01



(e) γ =0.05



(f) γ =0.1

圖 4. 10：上行 256Kbps,下行 8Mbps, γ 變動下不同傳輸協定效能之比較

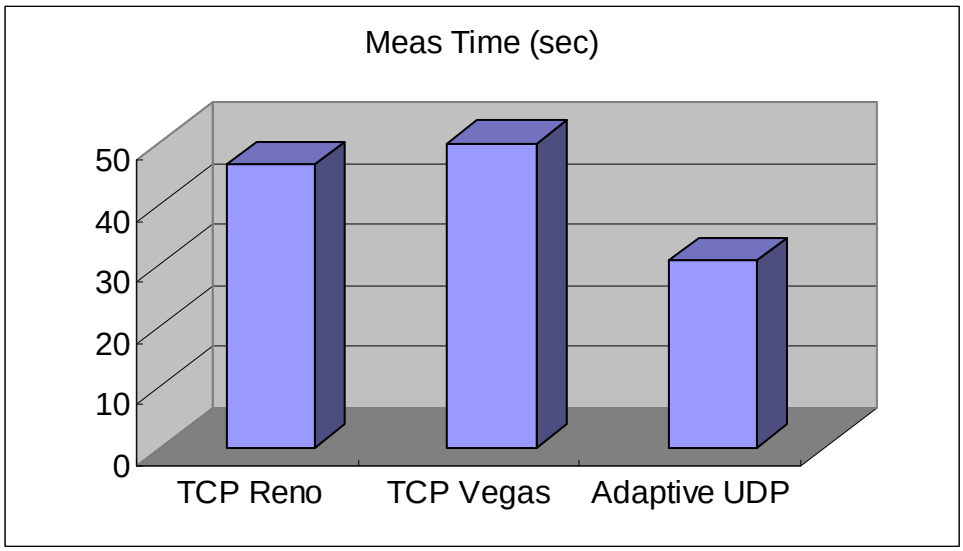


圖 4. 11：TCP Reno、TCP Vegas、Adaptive UDP 完成 1/4MB 資料之平均時間

表 4. 4 : TCP Reno、TCP Vegas、Adaptive UDP 完成 1/4MB 資料之效能比較

時間(sec)	TCP Reno	TCP Vegas	Adaptive UDP
Mean	46.5	50	30.8

TCP Reno 每個 Session 平均要花 46.5 秒，TCP Vegas 每個 Session 平均要花 50 秒，而 Adaptive UDP 每個 Session 平均只要花 30.8 秒，所以由上表的資料可以看到，Adaptive UDP 的資料傳輸效能明顯的優於 TCP Reno 及 TCP Vegas。

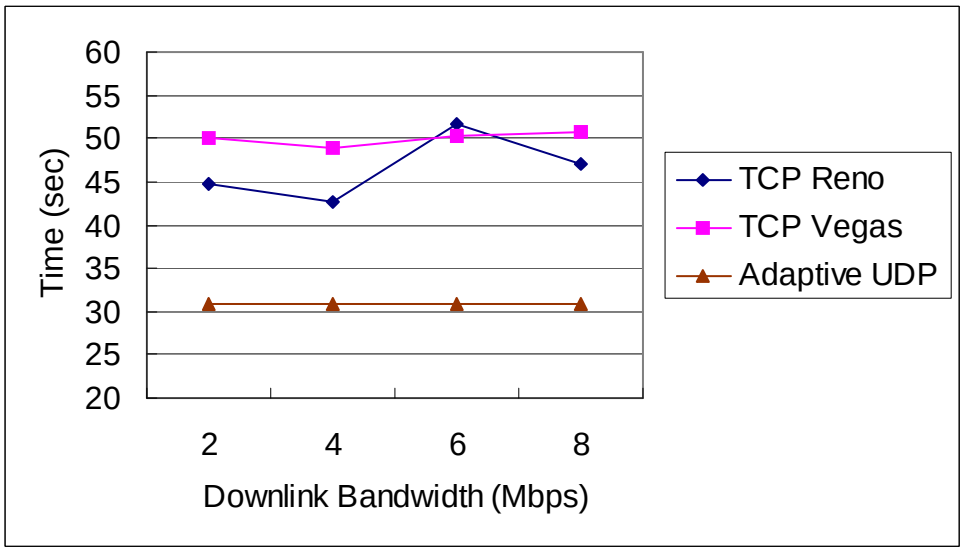


圖 4. 12：TCP Reno、TCP Vegas、Adaptive UDP 不同下行頻寬之敏感度分析

由圖 4.12 看到，固定上行頻寬為 256 Kbps，各個協定在下行頻寬 2Mbps～8Mbps 變動之下，下載完成時間變化皆不大，顯示增大下行頻寬並不能對 BitTorrent 效能提昇有所幫助。

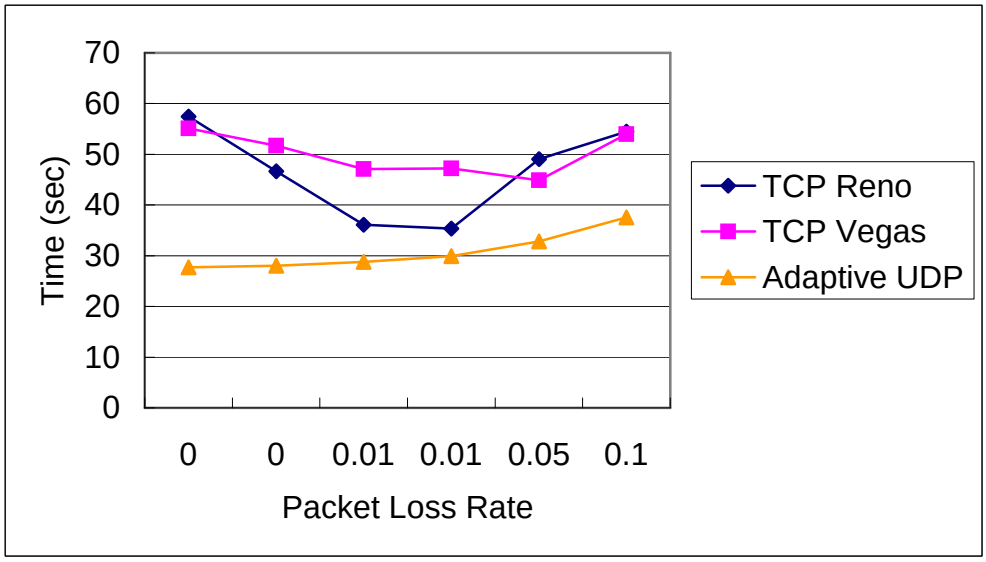


圖 4. 13：TCP Reno、TCP Vegas、Adaptive UDP 不同封包遺失率之敏感度分析

由圖 4.13 看到，各個協定在增加封包遺失率時，其下載完成時間有往上昇之趨勢。TCP Reno 與 TCP Vegas 若在網路擁塞時，有些許 Packet Loss 有助於效能之提昇。

還有一點需要特別拿出來討論，關於 TCP Reno 在非對稱網路之下資料傳輸的表現(目前實際網路上最常見的 TCP 版本是 TCP Reno)。由上圖中，可以明顯看到 TCP Reno 的效能表現相當不好，在 TCP Reno 的實驗中，確實發現有些 TCP Reno 的資料因擁塞被 Block 至逾時 (BoA)，而導致傳送端重新再傳一次此逾時資料，以及 ACK 因為回傳的上行過窄而被 Drop 掉的情形。這些 FUB 與 BoA 所引起的意外，都會導致 TCP Congestion Window 的縮小，所以最後造成了 TCP Reno 的嚴重效能不彰。

TCP Vegas 是利用 RTT 的時間來調節傳送速率，因為它比較不依賴封包遺失的訊號(Packet Lost)來調節資料傳送速度，所以封包遺失率相較來說較低，所以 ACK 封包遺失對其效能造成之影響不致於太大。但是，由於 TCP Vegas 不能分辨在非對稱網路下傳輸方向的不同，而導致其調節後上行和下行之速率大致相等，白白浪費上寬裕的下行頻寬。

4.3 總結

由以上的實驗與分析，增大下行頻寬並不能對 BitTorrent 效能提昇有所幫助。而 TCP Vegas 不能分辨在非對稱網路下傳輸方向的不同，而導致其調節後上行和下行之速率大致相等，其表現反而不如更早提出之 TCP Reno。而且我們發現，使用 UDP-Based Approach 有不錯的效能，而且此法的優勢在於設計簡單，不需對底層協定大幅改版即有不錯的表現。這些優點，相信對 BitTorrent-like 系統之使用者會有很大的吸引力。

第五章

結論

5.1 結論

在這篇文章中，我們對 P2P 檔案共享系統在非對稱網路下，其效能不佳進行分析，由於其中 Blockage of ACK 的問題目前並沒有研究學者提出完整的解決方案，所以，我們針對問題，設計了一個傳輸層的網路協定進行改善。我們所設計之協定以 UDP 為基礎，並且在應用層上加上自動重建遺失之封包、決定基本傳輸單位大小及決定資料傳送速率等配套措施。協定設計的過程中，建立了參數決定的數學模型以及利用模擬進行效能分析。結果發現，我們所設計的傳輸協定效能和 TCP Reno、TCP Vegas 比較之下，表現得還不錯。我們希望這個設計出來的網路協定能有效提昇 P2P 檔案共享系統的運作效能。

但是本協定需要 P2P 檔案共享系統內的傳送者與接收者同時採用，傳送端或接收端只要有一方不願配合，本協定即無法順利運作。要說服網路上為數眾多的 P2P 檔案共享系統參與者全數採用本協定，其推展上有一定難度，因此本研究的未來展望，希望可以考慮設計成只要資料接收端一方採用本協定之下，本協定即可順利運作，如此，只要下載者一方願意採用，即可有效提昇下載速度，更可提昇本協定之實用性。

Acknowledgement

感謝政治大學應用數學系陸行老師的協助，與資訊科學系行動計算實驗室所有老師與同學的大力幫助。

Reference

[1] H. Balakrishnan and V. N. Padmanabhan, "How Network Asymmetry Affects TCP," IEEE Communication Magazine, Vol. 39, No. 4, April. 2001, pp. 60-67.

[2] B. Cohen, “Incentives Build Robustness in BitTorrent,” http://www.bitorrent.com/, May. 2003.

[3] C. Dovrolis et.al., “Packet-Dispersion Techniques and a Capacity-Estimation Methodology,” IEEE/ACM Transactions on Networking, Vol. 12, No. 6, December 2004, pp. 963-977.

[4] Wanjiun Liao and Yi-Der Li, "Improving TCP Performance for Asymmetric Networks," IEEE ICC, Helsinki, Finland, June 2001, pp. 1824-1828.

[5] Yao-Nan Lien, "Performance Issues of P2P File Sharing Over Asymmetric and Wireless Networks," Proceedings of The First International Workshop on Mobility in Peer-to-peer Systems (MPPS05), June 2005, pp. 850-855.

[6] L. Peterson and B. Davie "Computer Networks, A Systems Approach, 3rd Edition," The Morgan Kaufmann Series in Networking, May 2003.

[7] P. Prasad et.al., “Bandwidth Estimation: Metrics, Measurement Techniques, and Tools,” IEEE Network Magazine, Vol. 17, No.6, November/December

2003, pp. 27-35.

[8] B. Melander, M. Bjorkman, and P. Gunningberg, “A New End-to-End Probing and Analysis Method for Estimating Bandwidth Bottlenecks,” IEEE Global Internet Symp., 2000, pp. 415-420.

[9] M. Izal, et.al., "Dissecting BitTorrent: Five Months in a Torrent’s Lifetime," Proceedings of PAM, Antibes Juan-les-Pins, France, April 2004, pp. 415-420.

[10] D. Qiu and R. Srikant. "Modeling and Performance Analysis of BitTorrent-like Peer-to-Peer Networks," Proceedings of ACM Sigcomm, Portland, OR, Aug. 2004, pp. 367-377.

[11] H. Balakrishnan and V. N. Padmanabhan, “How Network Asymmetry Affect TCP,” IEEE Communications Magazine, Vol. 39, No. 4, April 2001, pp. 60-67.

[12] 鍾永彬與連耀南，"路由器輔助的TCP擁塞控制技術之設計," National Chengchi University, Master Thesis, September 2005.

[13] Stephanos Androutsellis-Theotokis et. al., "A Survey of Peer-to-Peer Content Distribution Technologies," ACM Computing Surveys, Vol. 36, No. 4, December 2004, pp. 335–371.

[14] I. Foster, and A. Iamnitchi, ”On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing,” Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03). Berkley, CA.

[15] http://www.bittorrent.com/

[16] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "Exploring the Design Space of Distributed Peer-to-peer Systems: Comparing the web, Triad and Chord/CFS," Springer-Verlag Lecture Notes in Computer Science, Vol. 2429, March 2002,

pp. 214-224.

[17] S. M. Lui and S. H. Kwok, "Interoperability of Peer-To-Peer File Sharing Protocols," ACM SIGecom Exchanges, Vol. 3, No. 3, August 2002, pp. 25-33.

[18] S. Iren et al., "The Transport Layer: Tutorial and Survey," ACM Computing Surveys, Vol. 31, No. 4, December 1999, pp. 360-404.

[19] D. D. Clark and D. L. Tennenhouse, "Architectural Considerations for a New Generation of Protocols," ACM SIGCOMM Computer Communication Review, Vol. 20, No. 4 , September 1990, pp. 200–208.

[20] H. Schulzrinne et al., "RTP: A Transport Protocol for Real-Time Applications," RFC 3550, July 2003.

[21] J. Postel, "Transmission Control Protocol," RFC 793, September 1981.

[22] V. Jacobson, "Congestion Avoidance and Control," Proceedings of SIGCOMM Symposium on Communications Architectures and Protocols, August 1988, pp.314–329.

[23] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", RFC 2001, Jan 1997.

[24] W. Stevens, M. Allman and V. Paxson, "TCP Congestion Control", RFC 2581, April 1999.

[25] V. Cerf and R. Kahn, "A Protocol for Packet Network Intercommunication", IEEE Transactions on Communications, Vol. COM-22, No. 5, May 1974, pp. 637-648.

[26] J. Postel, "Internet Protocol," RFC 791, September 1981.

[27] L. Brakmo, S. O'Malley, and L. Peterson, "TCP Vegas: New Techniques for Congestion Detection and Avoidance," Proceedings of the SIGCOMM '94 Symposium, August 1994, pp. 24-35.

[28] S. Floyd, "A Report on Recent Developments in TCP Congestion Control," IEEE Communications Magazine, Vol. 39, No. 4, April 2001, pp. 84-90.

[29] C. P. Fu and S. C. Liew, "A Remedy for Performance Degradation of TCP Vegas in Asymmetric Networks," IEEE Communications Letters, Vol. 7, No. 1, Jan 2003, pp. 42-44.

[30] G. Fox, "Peer-to-Peer Networks," IEEE Computing in Science and Engineering, Vol. 3, No. 3, May 2001, pp. 75-77.