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

連耀南　　許弘奇

國立政治大學資訊科學系

{lien, g9231}@cs.nccu.edu.tw

摘要

近年來分散式 P2P 檔案共享系統因可擴張性較佳而廣為風行。但是，在非對稱網路之下，受到兩個問題之影響，其效能受到很大的限制：(1) Fractional Upward Bandwidth (FUB)與 (2)Blockage of Acknowledgement (BoA)。本研究之目的，是要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。我們改用 UDP 作為傳輸層協定，並在應用層加入封包等級錯誤更正(Packet Level Error Correction)及決定資料傳送速率(Date Rate Determination)等機制，以彌補 UDP 之缺陷。文中並提供傳輸協定運作時所需參數的估計法，並且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。

關鍵詞：網路協定、P2P 檔案分享系統、分散式計算

Abstract

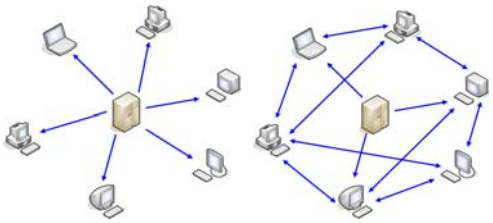
Distributed Peer-to-Peer (P2P) sharing system are widely used due to its scalability. Unfortunately, P2P file sharing system has several shortcomings on performance over asymmetric networks, due to two main problems: Fractional Upward Bandwidth and Blockage of Acknowledgement. We propose a new UDP-based protocol to alleviate the problems. Experiments show that our proposed protocol can improve its performance.

Keywords: Network Protocol, P2P File Sharing System, Distributed Computing.

1. 前言

P2P 架構允許社群內的使用者互相共享資源，例如資料內容、儲存空間、網路資源、CPU 計算時間等，使用者可享受它原本無法負擔的資源[6]，故而可擁有較大之資料儲存空間、舉行多人同時參與的視訊會議、進行較複雜的搜尋與計算等。其中最廣為風行的 P2P 系統當屬檔案共享系統 (File Sharing System)，如 BitTorrent。

P2P 檔案共享系統內，參與的角色分別有：(1)資料提供者及(2)資料下載者(Downloader)；而運作架構則有集中式及分散式兩種：有一共同資料來源，供其他人抓取檔案者稱為集中式架構(如 Napster)，當下載者太多時，系統無法負荷。反之，則為分散式架構(如 BitTorrent)，下載資料者亦可幫忙提供已下載之資料供他人下載。分散式架構可供較多人同時下載。我們的研究，將著重於探討 BitTorrent-like Model 的相關議題。



(a)　　(b)

圖 1　P2P 檔案分享架構 (a)集中式 (b)分散式

1.1 BitTorrent-like Model

由於 BitTorrent-like System 目前最為風行且擴張性極佳，可作為分散式 P2P 檔案分享系統之代表，我們以其作為研究對象。在[2]這篇文章中，載有 BitTorrent 協定的運作方式：資料提供者及下載者都須安裝 BitTorrent 軟體；原始檔案提供者要將其檔案公佈出來時，可利用 BitTorrent 程式建立 .torrent 檔，並指定「Tracker」的 URL，公佈在某網頁上；使用者在進行檔案下載前，至網頁伺服器下載 .torrent 檔，再利用 BitTorrent 開啟此.torrent 檔，經由 Tracker 來找到其他下載者，即可進行檔案下載； .torrent 除了有 Tracker URL 位置之外，還包含了要被下載檔案的相關資訊，如檔名、大小、檔案特徵(Signature)等訊息。而有完整檔案的下載者稱為 Seed。

BitTorrent 中有兩個基本檔案單位：Fragment (Piece)及 Sub-fragment (Sub-piece)。Fragment 之長度為 1/4 Mbyte，是檔案傳輸之基本單位；Sub-fragment 之長度為 16Kbytes，是利用 Pipeline 方式提昇 Fragment 傳輸速度之單位。而傳輸協定則選用 TCP。作者亦說明如何選取要下載的 Fragment，以及作者精心設計的 Choking Algorithm 用以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。

1.2 TCP 簡介

TCP 於傳輸資料之初，利用 Slow Start 機制探測目前網路可承載的頻寬。之後，再利用 AIMD (Additive Increase/Multiplicative Decrease)的方式進行擁塞控管。TCP 以封包遺失為網路擁塞的指標，資料傳輸中若有封包遺失，TCP 會啟動擁塞控制機制快速降低傳輸速率。

1.3 P2P 檔案共享系統之效能缺陷

雖然 P2P 檔案共享系統已非常風行，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。一個值得深思的問題是：「當擁有寬鬆的下行頻寬時，為何下載速度很慢？」

1.4 P2P 檔案共享系統效能問題之分析

經由我們的探討， P2P 檔案分享系統效能不彰問題之部份成因如下：

(1)Fractional Upward Bandwidth (FUB) 的問題：一部電腦內的檔案分段可同時被多個下載點同時下載，但因 ADSL 上傳的頻寬窄，導致多個上傳串流要共用一個較窄的頻寬。

(2)Blockage of Acknowledgement (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，較窄的上行頻寬擁塞使得 ACK 無法順利回傳，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，我們可將檔案資料所有下載者之間所形成的關係想像如圖 2：同一個檔案片段(File Fragment)以 Seeder 為樹根(Root)，其資料上傳者和下載者以 link 連接，如此形成一個階層架構，我們稱之為檔案片段樹(Fragment Tree)。

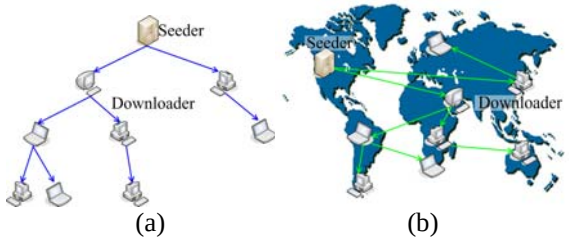


圖 2 Fragment Tree (a)Logical Fragment Tree (b)Physical Fragment Tree

由檔案片段樹可以觀察到下列結果：

(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(path)，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。如果能盡量縮短路徑、減少重覆，必能降低頻寬之浪費。

(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，有人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]。

在非對稱網路上 BitTorrent-like 系統之效能無法提昇，經由之前的分析可得知，是受 FUB (Fractional Upward Bandwidth) 與 BoA (Blockage of ACK)的影響。本研究的目的，即針對此效能問題提出改進措施。

1.5 論文組織架構

本文第一章簡介 BitTorrent-like Model 在非對稱網路上使用 TCP 協定時所遭遇的各種問題；第二章對非對稱網路協定之問題的文獻作簡介和評論；第三章介紹本研究所提出之「UDP 為基礎之協定」的有限狀態機運作流程及其細部設計；第四章展示本研究之協定參數計算結果並以模擬方法評估我們提出協定之運作效能；第五章為本文結論與未來展望。

2. 相關研究

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。其中，TCP Vegas 係經由觀測 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，根據此資訊控制傳輸速率可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下因傳輸方向不同所產生的負面影響，導致整個效能大為降低。此文提出一個新的 TCP Formosa 協定。為解決非對稱網路問題，此協定設計了一個 Cumulative ACK 的機制，減少原來傳輸時 ACK 之數量，避免非對稱網路之下因為 ACK 遺失所導致的影響，此協定在一定程度上解決了非對稱網路 TCP 之效能問題，獲得不錯的成果。

3. 協定設計

有數個方案可以解決 BoA 所導致的效能降低問題：

方案一：增加 TCP ACK Timeout 的時間。但此法有一副作用，因為 TCP 是利用 Timeout 的時間決定一個封包是否遺失，增加 TCP ACK Timeout 的時間時，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)。

方案二：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，可能惡化 BoA 問題。

本文提出方案三：以 UDP 為基礎的方式(UDP-Based Approach)解決。假設傳送者可測得實際可用頻寬(Actual Available Bandwidth)之大小，本法使用 UDP (UDP-Based Approach)傳送資料。在 UDP 協定中，接收端不需要對接收到的封包回送 ACK，因此可以避免 BoA 問題，讓傳送端不會無謂的降低傳送速度。但 UDP 協定有兩個問題：(1)無法確保資料的完整性。(2)無法自動決定資料傳送速率。為此，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。

為了提高效能，必須作到 (1)儘量避免重傳次數及(2)儘量利用可利用之網路頻寬，提高吞吐量(throughput)。故此協定中，包含特別三項與傳輸效能相關的運作機制，其分別說明如下：

(1)Packet Level Error Correction(封包等級錯誤更正)： 為避免封包遺失引發太多的檔案片段重傳，我們加入自動錯誤更正機制。

(2)Fragment Size Determination(檔案片段單位估算)：計算檔案片段中，最佳的基本傳輸單位大小，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，調整傳送速度。

3.1 Packet Level Error Correction

為了避免封包損傷導致太多的檔案片段重傳，我們設計了一個封包等級的錯誤更正機制(Packet Level Error Correction)。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，稱為一個自保封包組(Self-Protective Packet Set)，而 n 值稱為自保封包數。

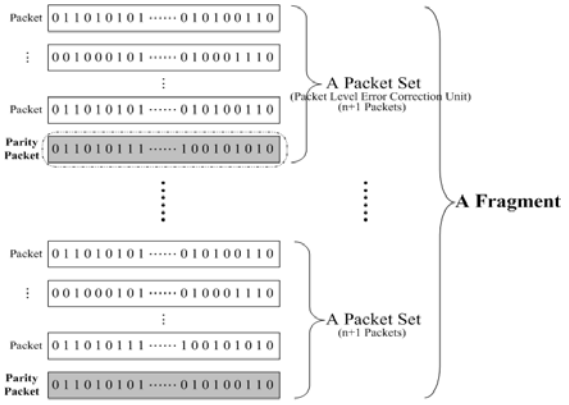


圖 3 Packet Level Error Correction

當傳輸一個檔案片段時，任一自保封包組中若有兩個以上封包出現錯誤，同位封包將無法彌補，此檔案片段必須重傳。接收端提出重傳受損檔案片段之請求，要求傳送端再發送一次該檔案片段。其細部運作流程如圖 4 所示：

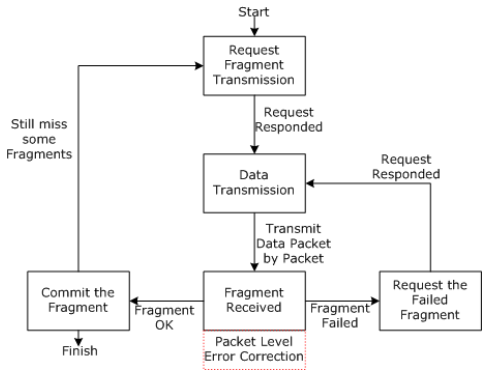


圖 4 Packet Level Error Correction

自保封包數之大小會影響協定之效能，其值較小時，錯誤更正能力較強，但 Overhead 變大。其值較大時， Overhead 較小，但錯誤更正能力也相對較弱。最佳自保封包數之決定將在下節中討論。

3.2 Fragment Size Determination

如圖 5，檔案結構定義如下：

(1)一個檔案可切成 k 個檔案片段(Fragment)。

(2)一個檔案片段，總共有 m 個封包 (未加上同位封包時)。

(3)一個自保封包組內含 n+1 個封包。一個檔案片段是由一個以上的自保封包組組合而成。

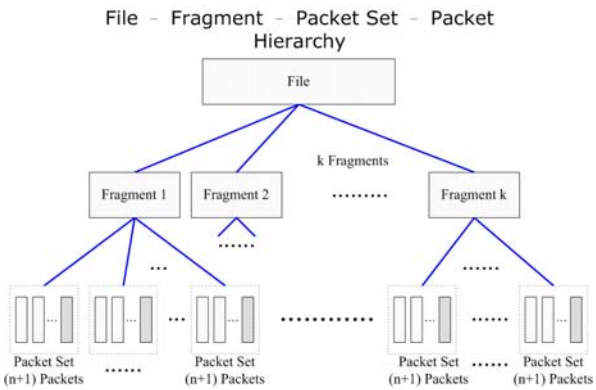


圖 5 檔案層級架構

上節曾提及自保封包數之大小會影響傳輸效率。本節介紹我們所設計尋找最佳自保封包數之估算法，所定義之符號如表一。

表 1 自保封包數 n 值估算之符號定義

符號	意　　義
m	一檔案片段(Fragment)中之封包數
γ	封包遺失率, $0 \le \gamma \le 1$
l	封包大小 (單位：byte)
n	一個自保封包組之封包數

我們推導一個最佳化模型。以 BitTorrent-like 檔案共享系統之最佳傳輸效率為最佳化目標，檔案片段之封包數、封包遺失率、封包大小為已知參數。

給定 m, r, l
找到 n
使得 Penalty Function 值為最小
限制條件為
檔案大小=檔案片段之封包數x封包大小

(1) 一個封包組中，x 個封包遺失的機率

一個封包組的 $n+1$ 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) 一個封包組傳送成功的機率

因為我們設計錯誤更正機制的單位為封包組。一個封包組，可救回一個遺失或錯誤的封包。所以，一個封包組的成功機率為「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) 一個檔案片段的失敗機率

一個檔案片段總共可以分成 $\left\lceil \frac{m}{n} \right\rceil$ 個封包組。此 $\left\lceil \frac{m}{n} \right\rceil$ 個封包組全部傳送成功的機率為：

$$\left(bin(0 \mid n+1, \gamma) + bin(1 \mid n+1, \gamma)\right)^{\left\lceil \frac{m}{n} \right\rceil} \qquad (3)$$

反之，一個檔案片段傳送失敗的機率為：

$$\left[1 - \left(bin(0 \mid n+1, \gamma) + bin(1 \mid n+1, \gamma)\right)^{\left\lceil \frac{m}{n} \right\rceil}\right] \qquad (4)$$

(4) 額外成本

一個檔案片段須額外傳送 $\left\lceil \frac{m}{n} \right\rceil$ 個同位封包。其傳送量為：$\left(\left\lceil \frac{m}{n} \right\rceil \times l\right)$。

當一個檔案片段傳送失敗時，整個檔案片段須重傳一次，其重傳成本為：$\left((m + \left\lceil \frac{m}{n} \right\rceil) \times l\right)$。

總額外成本我們稱之為懲罰函數(Penalty Function)為：

$$Penalty(n \mid m, \gamma, l) = \left((m + \left\lceil \frac{m}{n} \right\rceil) \times l\right) \times \left[1 - \left(bin(0 \mid n+1, \gamma) + bin(1 \mid n+1, \gamma)\right)^{\left\lceil \frac{m}{n} \right\rceil}\right] + \left(\left\lceil \frac{m}{n} \right\rceil \times l\right)$$

, $1 \le n \le m$. (5)

(5)當 $Penalty(n \mid m, \gamma, l)$ 最小時，可得最佳 n 值。因 n 值之範圍不大($1 \le n \le m$)，故可用 linear search 方式快速求得最佳 n 值。

3.3 Adaptive UDP Mechanism

由於 UDP 協定自身沒有決定傳送速率的機制，我們必須在所設計之檔案分享協定中加入自動調整速率機制。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)。路徑中瓶頸之發生點有二：(i)發生在使用者上行端，或者(ii)發生在網路內部。為處理瓶頸發生於網路內部之情況，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之狀況。

傳送端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所得之最適速率傳送資料，可以降低頻寬浪費或網路擁塞之機率。

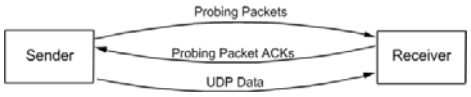


圖 6 UDP with Probing Packet

緊鄰送出的兩個封包，在無其它因素干擾下，這兩個封包在通過瓶頸時，會因為瓶頸網路處理速度不夠快，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[3]。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：

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

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，進行資料傳送速率(Data Rate)的調整。

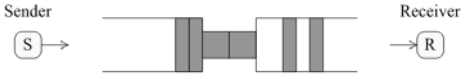


圖 7 Packet Dispersion

為避免估計偏差(bias)對我們協定造成負面影響，我們使用修正係數 α (Network State Adjusting Coefficient α)用以效正估計結果。其效正公式如

下：

$$Estimated\_Bandwidth(bytes/sec) = \alpha \times \frac{Packet\_Size(bytes)}{Average\_Dispersion\_Time(sec)} \quad (7)$$

最後比較上行頻寬(Uplink Bandwidth)和所測得之估計頻寬(Estimated Bandwidth)，取其最小值，做為傳送速率之參考。

4. 參數估算及效能評估

本研究所提出之網路協定，檔案片段傳送單位之決定需要進行參數估算。最後，我們再以 UDP-Based Approach 為實驗組，以 TCP-Reno、TCP-Vegas 為對照組，進行效能評估比較。

4.1 檔案片段傳送單位之參數估算

實驗目標：給定特定之網路環境，利用最小化懲罰函數找出自保封包數，並且觀察在不同封包遺失率下，自保封包數之變化情形。

計算範例所設定之實驗參數為：

表 2 尋找最佳自保封包數之實驗參數

參數	數 值 範 圍
m	250 packets/fragment
γ	0.005~0.02
l	1000 bytes/packet

經過 Matlab 計算後，可得到如下之關係圖：

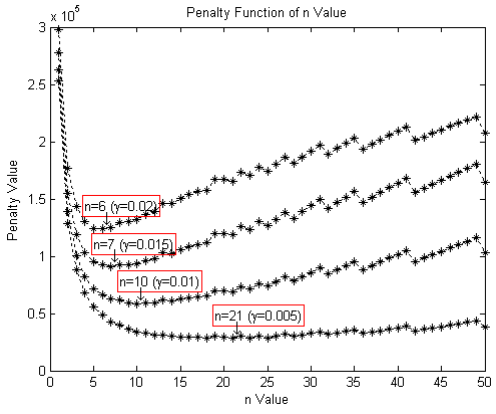


圖 8 Penalty Function of n(γ =0.005, 0.01, 0.015, 0.02)

圖 8 中，橫軸為自保封包數的變動，縱軸為 Penalty Value。如 γ =0.005 時，一個封包組之自保封包數為 21 時，可得到最小 Penalty Value 為 28492.4。

其次，我們想研究自保封包數在封包遺失率不同之下之敏感度。

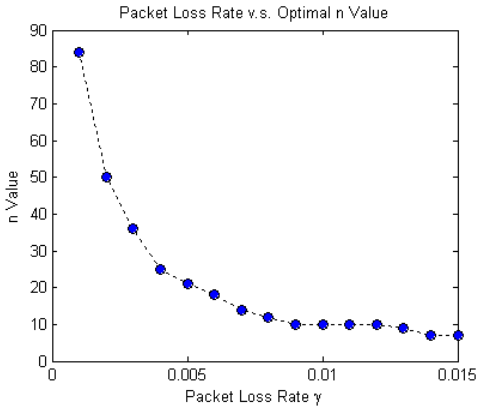


圖 9 自保封包數 n 值對封包遺失率之敏感度分析

由上圖 9 可以看到，在封包遺失率(Packet Loss Rate)很低之下，求得之自保封包數較大，表示此時資料傳輸時，並不需太多的同位封包以保護資料封包。如果封包遺失率提高時，最佳自保封包數會較小，表示此時需較多之同位封包來保護資料封包。所以，圖 9 所示自保封包數對應封包遺失率之趨勢圖，符合預期結果。

4.2 UDP-Based Approach 與其他傳輸協定之效能比較

實驗目標：設計一個會發生 FUB 與 BoA 之非對稱網路 P2P 檔案分享系統的場景，並且對各種不同的傳輸協定進行效能分析比較。

實驗設計：實驗的場景設定在核心網路(Core Network)中有充沛頻寬，雙向頻寬為 100 Mbps，而非對稱網路之上下行頻寬為 56 Kbps 及 8 Mbps。每個分享檔案片段為 1Mb。我們設計一非對稱網路之網路拓樸如圖 10，其中 X1～X5 同時會至 Y1 下載資料，而且 Y1 亦有 5 個 Session 同時至 X1～X5 下載資料，共十個 Session，在 Y1 至 R2 之上行鏈結發生 FUB 與 BoA 的問題，我們即可量測在不同的傳輸協定之下，對 P2P 檔案分享系統之效能。實驗之模擬平台，採用 Cygwin 下之 ns 2.28 版。

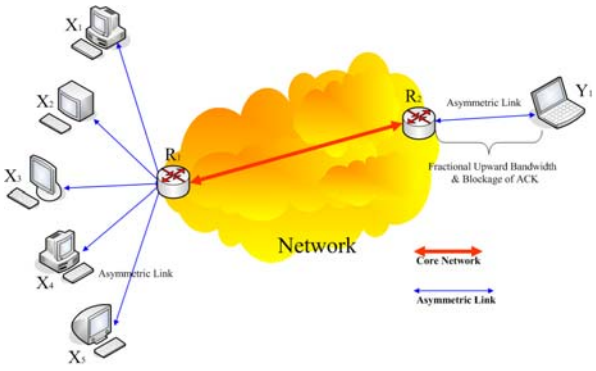


圖 10 Simulation Topology

實驗結果與分析：

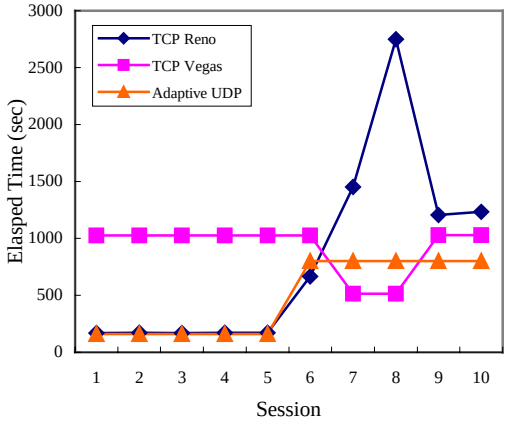


圖 11 TCP Reno、TCP Vegas、Adaptive UDP 完成 1MB 資料之效能比較

TCP Reno、TCP Vegas 及 UDP-Based approach 完成傳輸一個 session 的時間分別為 815.4 秒、924.2 秒、479.8 秒，所以 Adaptive UDP 的資料傳輸效能明顯的優於 TCP Reno 及 TCP Vegas。

由上述數據，可看到 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 的資料因擁塞被堵塞導至 Timeout，以及 ACK 因為回傳的上行過窄而被丟棄。這些 FUB 與 BoA 所引起的意外，都會導致 TCP 擁塞窗框(Congestion Window)縮小，最後造成 TCP Reno 的效能不佳。特別是 TCP Reno 在 Session 8 的完成時間相當長，我們推測此乃 Session 8 發生了相當程度的封包遺失之後，其資料傳送速率之回復已難追上其他 TCP Reno 之 Session。

TCP Vegas 是利用 RTT 的時間來調節傳送速率，較不依賴封包遺失的訊號來調節資料傳送速度，但因 TCP Vegas 無法分辨非對稱網路之方向性，其上行與下行之速率調節後大致相等，未充分使用充裕之下行頻寬，故其上行與下行之 Session 之完成時間相當接近。

5. 結論

本文中，我們對 P2P 檔案共享系統在非對稱網路下進行分析，並且針對分析所得到的問題，設計一個傳輸層網路協定改善其效能。協定設計的過程中，建立了參數決定之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表現優於 TCP Reno 及 TCP Vegas。希望我們所提出之網路協定能有效提昇 P2P 檔案共享系統的運作效能。

參考文獻

[1] Hari Balakrishnan and Venkata N. Padmanabhan, "How Network Asymmetry Affects TCP," IEEE Communication Magazine, Apr. 2001.

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

[3] Constantinos Dovrolis et.al., “Packet-Dispersion Techniques and a Capacity-Estimation Methodology,” IEEE/ACM Transactions on Networking, Dec. 2004.

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

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

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

[7] Pavi Prasad et.al., “Bandwidth Estimation: Metrics, Measurement Techniques, and Tools,” IEEE Network Magazine, Nov./Dec. 2003.