(0-3

利用策略式動態拓樸調整來降低類 BitTorrent 系統之平均下載時間

Chun-Hsien Lu（呂俊賢）and Yi-Syuan Hung（洪翊軒）
Dept. of Computer Science and Information Engineering
Fu Jen Catholic University
輔仁大學資訊工程系
jonlu@csie.fju.edu.tw

Abstract BitTorrent has been the most popular P2P file sharing system, in which a peer upon joining the system will be given a candidate neighbor list to set up neighboring connections, and those connections will not change till the end of file downloading. This situation brings some problems. For example, if the high-bandwidth and low-bandwidth peers are allocated in the neighbors unevenly, high-bandwidth peers may quickly complete file downloading and leave the system, while low-bandwidth peers are left behind to download the pieces slowly in the system. We propose an algorithm called Strategic and Dynamic Topology Adjustment (SDTA) to reduce average download time in BitTorrent-like system. When a peer is downloading inefficiently, it will try to add a new neighbor based on its own degree of completion and uploading bandwidth. Our goal is to let high-bandwidth peers reach a high completion state, so that they become helping peers to serve low-bandwidth peers. In addition, adding a new neighbor can help reduce waiting time when a peer cannot find a desirable piece among its neighbors. Simulation results show that our SDTA algorithm can produce a 13% to 28% percent reduction over BT in the average download time.

Keywords: BitTorrent, Topology adjustment, Request Selection

摘要 — 在 BitTorrent 系統中，使用者會透過 tracker 給予的鄰居清單來建立節點之間的連結，而且確立連結後即不會做拓樸的更動，假設拓樸在建立時高低頻寬的鄰居分配不均，便容易造成高頻寬的節點快速完成檔案下載後離開系統，而低頻寬的節點由於其下載及上傳能力皆相形見絀，而長時間延宕在系統中。為了改善上述問題，我們提出一個 Strategic and Dynamic Topology Adjustment (SDTA)演算法，針對不同頻寬與完成度的使用者來給予特定的新鄰居清單。其精神在於希望透過拓樸的更動，讓鄰居圈中高低頻寬的節點分佈變得平均，並且讓高頻寬節點能夠扮演能者多勞的角色。我們將可能的情況概括分為三個策略，首先是策略二會優先讓高頻寬節點之間透過新增鄰居機制來互相得到大多數的片段，使其成為幫助者，之後改進行策略一去尋找低頻寬的節點來給予幫助。而最後的策略三則是將低頻寬節點視為需要幫助者，不斷去尋求高頻寬鄰居的幫助。實驗結果顯示我們的 SDTA 方法在檔案平均下載時間方面，相較原 BT 方法可以有 13%~28%的改善。

關鍵詞： BitTorrent、拓樸變動、服務策略

1. 導論

在資訊爆炸的時代，透過網際網路實現「秀才不出門，能知天下事」的概念已不是天方夜譚。然而隨著使用者對於資訊需求量的成長，網路傳輸耗費量亦直線飆漲。傳統主從式架構雖然具有方便管理及維護等優點，但是如果在短時間內收到用戶端過量的連線要求，便會造成伺服器無法負荷，進而產生網路傳輸的壅塞，大幅增加用戶端的下載時間，使得服務效率每況愈下。點對點網路[1-9]是解決此問題的有效方案，在點對點架構下，每個使用者通稱為節點(peer)，同時擔任著用戶端及伺服器端的角色，如圖 1 所示。每個節點在下載的同時，也會貢獻出自己的上傳頻寬來傳輸檔案給其他的節點，如此達到相互幫忙，資料快速擴張的效果。

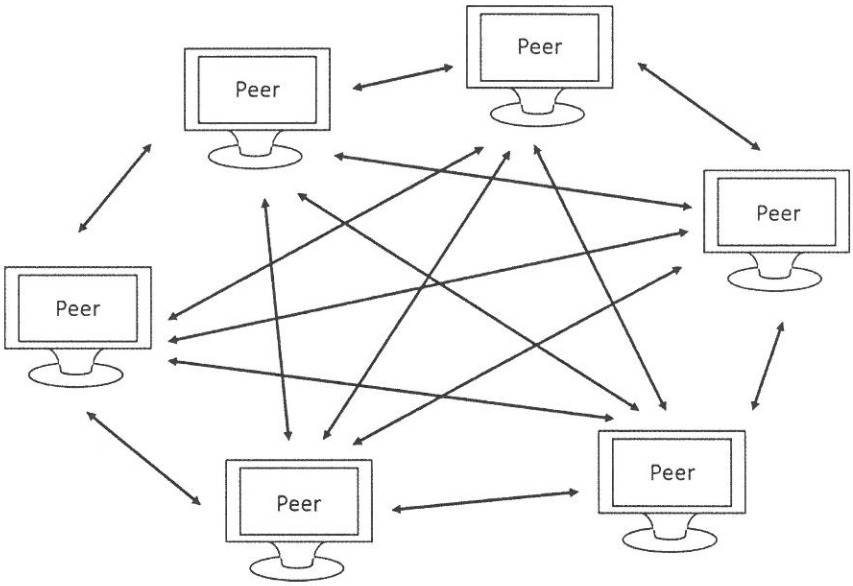

圖 1 、點對點系統架構

檔案下載是點對點架構中最為被討論的重點之一，而早期的點對點下載，節點必須完成檔案下載後才能再分享給其他的節點。但許多節點可能在下載完成後會立即離開系統，造成這些節點只下載而不上傳，此現象稱作 Free-Riding。BitTorrent[1]協定的提出能有效解決上述問題，它將檔案分割成數個等份片段，只要完成其中一個片段的下載，就能夠開始分享該片段。但直到獲得所有片段，節點才能夠組成完整檔案進而離開系統。這讓節點分享檔案的時間大幅增加，有效防止 Free-Riding 的情況。當節點加入 BT 系統後，會獲得由 Tracker 所提供的一份鄰居清單，

該節點可在清單中選取足量的節點與其建立連結，成為鄰居關係，此處的選取機制為隨機挑選。另外，除去新鄰居的連結加入或是鄰居檔案完成下載後離開系統之外，每個節點直到檔案下載完成，本身已建立的拓樸並不會做任何的更動。上述的情況衍生出兩個問題，其一為如果節點們選擇的鄰居頻寬分配不均，便會造成同為高頻寬的鄰居節點，快速相互幫助完成下載後離開系統，低頻寬的節點卻因為其下載能力較差，又沒有高頻寬的鄰居給予幫助而不斷延宕在系統內。其二則是無論是高頻寬或低頻寬的節點，在拓樸不變動的情況下，若鄰居皆在忙碌或是當下鄰居中皆沒有可以服務該節點的片段，造成發出的片段請求進入長時間的等待，下載頻寬因此浪費使得傳輸效率變差。

我們提出一個方法，讓節點在碰到下載瓶頸時能夠加入新鄰居，然後在達到鄰居數量上限時作條件式的切斷連結策略。新鄰居的選擇機制是考量本身節點的頻寬及完成度、目標新鄰居的頻寬及完成度，以及彼此間緩衝區的互補性或是貢獻性，綜合上述情況設計出一套演算法，希望透過調整鄰居拓樸的方式，降低系統內平均的檔案下載完成時間。

本論文其餘部分的內容如下，第二節為相關研究的探討，第三節是我們提出的方法策略及說明，第四節則是實驗模擬與效能評估，最後第五節為結論。

2. 相關研究

在降低 P2P 網路的平均下載時間，目前已有許多針對拓樸變動的學術論文被提出。其中[10]在類 Gnutella 的環境下，針對傳送的要求次數來變動鄰居拓樸。作者認為在使用廣播尋找目標檔案的情況中，會因為重複傳送或搜索跳數(Hop)過多等問題造成許多不必要的資源浪費，但在設置 TTL 後卻又無法保證搜索的成功。因此為了減少搜索的跳數及提高搜索成功率，進而提出 Topology Re-formation Algorithm，透過鄰居記錄節點之間過去一段時間內傳送的資料量，將彼此有大量資料傳送卻不是鄰居的兩節點新建為鄰居。此處被切斷連結的 P2 作類似的計算新增一個鄰居以維持鄰居個數不變。此方法雖然理論上能夠有效降低搜索跳數，但卻會造成拓樸頻繁變

動，致使原本發出的片段要求得一再重複發送，反而無法提升效能，因此拓樸變動頻率必須設定在一個適當的值。同樣在類 Gnutella 的環境下，[11]中作者提出的 Adapting Peer-to-Peer Topologies 方法係經由實驗統計觀察出在傳送廣播時，搜索的跳數越多，同時也會增加訊息重複收取並傳送的機率。為了減少此不必要的浪費，將其發生的數量換算成分數排名，並把其中重複最多次的節點切斷與原鄰居的連結，再將其重新加入系統。除此之外，新增鄰居來建立捷徑即能減少節點之間的搜索範圍，作者利用 Diameter Folding 的方法，將發送節點及已被廣播搜尋到的檔案持有者之路徑中所有節點進行距離計算，計算出兩者之間相對最短且對系統整體效益也不錯的路線，調整其拓樸。

[10]和[11]皆只考慮到節點間的實際連結距離而作出拓樸調整，然而對於節點之間的鄰居幫助通常不大，因此[12]考慮到原節點與新鄰居相互能夠互補所需的片段情況下，也能對兩方鄰居圈達到幫助的效果。BT 的鄰居拓樸是隨機建立而忽略了實體連結距離，因此常會有相當多跨越網際網路服務供應商(Internet Service Provider，ISP)的情形，傳輸效益和實際距離成反比導致無論對節點或對系統都是極大的負擔，因此另一種為降低整體下載時間而更動拓樸的方法即為降低跨 ISP 的流量。[12]中作者提出 Adaptive Complementary Neighbor Selection (ACNS)演算法，將在不同 ISP 的鄰居節點以互補程度及對對方鄰居圈的貢獻程度為依據，來計算出 PCI (Piece Complementary Index)指數。當 PCI 值小於一個門檻時，則視其跨 ISP 的鄰居為沒有互相幫助的效果，而切斷彼此連結，然後再透過 Tracker 從不同的 ISP 找尋新的節點來建立連結。

在點對點網路中，許多節點在完成檔案的下載後即離線，這些節點不願意耗費自己的上傳頻寬來分享片段給使用者，造成許多還在下載的節點失去片段來源，只能依靠 seed，因而失去點對點分享頻寬的概念。因此在公平性與效益性的取捨上，許多學術論文都會設法將有較大分享能力的高頻寬節點扣留在系統中，為了能讓他們作到能者多勞的效果，在效益上有相當的成果改善，作者[13]提出的方法 Delaying Completion of Fast Peers (DCFP)是利用修改 request scheduling 的服務策略，將高頻寬的節點分為高完成度狀態以及低完成度狀態。所有節點的服務

策略皆為高頻寬低完成度的節點最為優先，目的是希望將高頻寬節點快速達到高完成度的狀態。而一旦變成高完成狀態的高頻寬節點，其服務優先權則被降至最低，為了就是將其扣留在系統，運用其高頻寬節點的分享能力去幫助其它未完成的節點。DFCP 方法在系統後期，由於高頻寬高完成度的節點的服務優先權最低，所以低頻寬節點會較快完成而離開系統，平均頻寬因而隨之上升。在整體效益上，DFCP 優於 BT 方法約 7%~28%，可知有效應用高頻寬節點的分享能力，即可在平均完成時間獲得不錯的改善成績。我們認為，若能夠將新增的鄰居中也考慮到高頻寬節點的頻寬運用，加上前述[12]的 ACNS 機制中互補性以及貢獻程度的概念，便能更進一步優化原 BT 系統。

3. 方法設計

3.1 系統假設

我們假設一個類似 BitTorrent 的系統，所有節點會同時加入此系統，並透過非結構式的形式隨機建置拓樸架構。節點之間的頻寬能力各有不同，系統中另外存在一個 seed，擔任不離線的服務者(類似主從式架構中伺服器的角色)，優先服務能力較強的節點。為了減少節點對 seed 要求服務的數量，節點有較高的優先順序來向鄰居要求片段。我們假設每個節點一旦完成檔案下載，便會立即離開系統。

3.2 方法內容

我們提出一個 Strategic and Dynamic Topology Adjustment (SDTA)的方法來動態調整拓樸，以改善節點彼此間的下載效率。節點加入點對點網路後，利用 BitTorrent 的片段選擇策略-最稀少片段優先 (Local Rarest First)來逐一下載片段。每隔一段時間，節點就會去檢查這段時間內的下載量，是否符合自身頻寬能力可以獲得的片段量，依此來設定門檻值，判斷節點有無處於低效能的窘境，而需要去加入新鄰居。若確定要加入新鄰居後，節點會從鄰居清單中挑選出當下是鄰

居圈內前三大負載的節點作為媒介，此三大鄰居的鄰居們在此處成為初期的候選人清單，如圖 2 所示。挑選前三大負載的用意在於我們假設負擔大的節點在自己的鄰居圈處於一個疲於奔命的狀態，因此希望透過加入他的鄰居去幫他分擔服務要求。

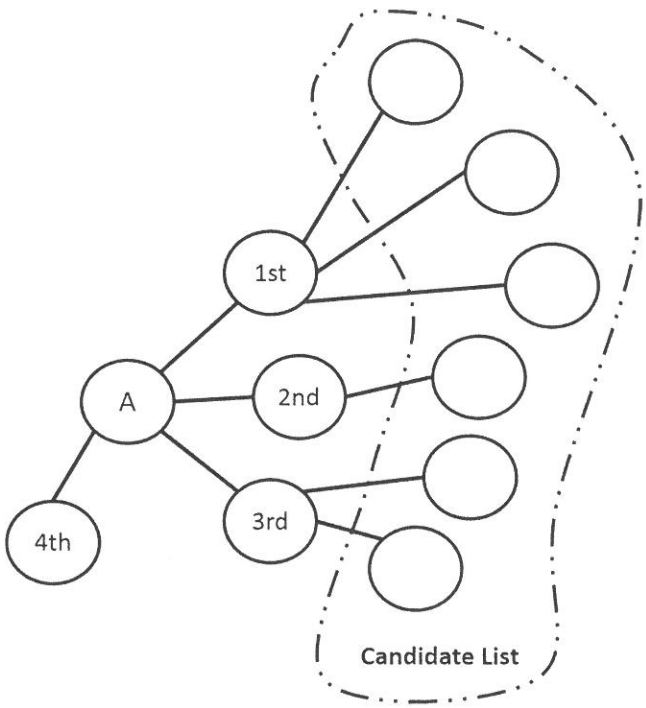

圖 2、初期候選人清單

有了初期的清單後，新鄰居候選人之篩選以節點自身頻寬及完成度，去判別目標新鄰居的頻寬、完成度，以及其片段貢獻程度(能夠服務我的片段量)或是緩衝區互補(能夠相互服務的片段量)程度等參數是否符合條件而定。前述狀況我們再細分為以下三種策略:

策略1: 當 A 是高頻寬且高完成度的節點，雖然 A 有較大的能力去獲取和服務片段，但因為我們假設節點完成後會立即離線，造成低頻寬的人在檔案下載後半段時間內缺乏除 seed 外的有力幫助者，所以我們希望能夠讓高頻寬的節點盡量被挽留在系統中，貢獻其上傳能力來服務低頻寬的節點，達到能者多勞的效果。因此，當節點 A 是高頻寬高完成度時，就會變成一個幫助者，去尋找頻寬較低且對節點 A 貢獻度也低的節點給予服務。低貢獻度是考量低頻寬節點較少的片段服務可讓高頻寬節點持續被扣留在系統中，且希望在低頻寬節點服務能力速度相對較慢的情況下，能夠盡量扣留住高頻寬節點。同時因為較低的貢獻，高頻寬

節點勢必會再去加入新鄰居，一樣是選取低頻寬少量貢獻的節點。此作法主要是想讓高頻寬節點可以貢獻服務給不同的低頻寬節點，逆向形成高頻寬節點作為幫助者去找尋需要被服務的人。

策略2: 當 A 是高頻寬但低完成度的節點，在下載早期，我們希望高頻寬節點之間可以相互服務，彼此快速進入高完成度的狀態，才能夠接續去幫助低頻寬的節點。因此，高頻寬且低完成度的節點會去挑選與自己情況相同的節點，並以彼此緩衝區中片段具有高互補性者為優先，為的就是可以填補相互缺乏的片段，達到快速擴張的效果。

策略3: 若 A 是一個低頻寬的節點，由於其頻寬能力較差，因此不論其完成度高低，都將他設定成一個需要被幫助的角色，去尋找高頻寬且對於節點 A 是高貢獻度的節點來獲取服務。選擇高貢獻度的原因是我們認為貢獻度越低的節點，通常為片段量較為稀少的高頻寬且低完成度節點，為了讓這些節點可以專注於執行策略 2，因此我們相對選擇高貢獻度的節點。而高頻寬高完成度的節點對於低頻寬的節點來說通常都擁有高貢獻度，另外也是為了呼應第一個策略，高頻寬節點會尋找低頻寬的節點來服務。

有了上述三個策略後，我們必須決定貢獻度以及互補性的大小如何去計算。如果加進來的新節點所擁有的片段不但是某節點所需要，同時也是原本鄰居圈所缺乏的，如此則能夠提升加入的效益。我們使用了下列的公式，考量貢獻度以及互補性大小來計算出貢獻分數及互補分數，用來篩選出最符合頻寬及完成度要求的目標新鄰居。我們定義 $S_{ba}(p)$為片段 p 是否為節點 b 能夠貢獻給節點 a 的片段，並用以計算貢獻分數 Con(b,a)，而 N_a則代表在節點 a 的鄰居圈中也缺乏片段 p 的鄰居數量。公式如下:

$$S_{ba}(p) = \begin{cases} 1, & if\ p \in b_{buffer}\ and\ p \notin a_{buffer} \\ 0, & otherwise \end{cases} \qquad (1)$$

$$Con(b,a) = \sum_{p \in b_{buffer}} (S_{ba}(p) * N_a) \qquad (2)$$

在貢獻分數的方面，首先會判斷候選人能夠提供節點 a 哪些片段，然後接著去看這些片段在 a 的鄰居圈的需求情況，如果同時也是 a 的鄰居圈所缺乏的，則該片段權重就變大。公式(2)中乘上缺乏此片段的鄰居數量(N_a)得到片段 p 的分數，最後將所有片段分數加總起來，即算出節點 b 對於節點 a 的貢獻分數。

接著，我們用 Com(b,a)來表示 peer b 對於 peer a 的互補分數，並定義 b_p 及 a_p 分別為節點 b 及節點 a 是否含有片段 p，公式如下:

$$Com(b,a) = \sum_{p=0}^{n}(b_p \ xor \ a_p) + Con(b,a) \qquad (3)$$

互補分數則是先判斷相互能夠給予的片段量，為同時考慮到自己本身鄰居圈的需求，因此再加入前述的候選人對於 a 及 a 的鄰居圈的貢獻分數。依照片段順序作片段間 exclusive or (xor)的運算，加總出互相可以幫助的片段量，再加上節點 b 對於節點 a 的鄰居圈貢獻分數 Con(b,a)，得到最後的互補分數。

我們認為，若為了保持整體拓樸的平衡，每當新增一個鄰居就搭配切斷一個原鄰居會讓拓樸本身頻繁地變動，效益上並不會提高。但不斷的增加鄰居數量，最後可能變成系統中所有節點皆互為鄰居，反而不切實際。因此我們限制節點鄰居數量，當超過鄰居的最大數量時，便會去切斷鄰居的連結。切斷的策略是節點去計算當下鄰居圈中能夠給予該節點最少片段(貢獻度最低)的節點優先切斷連結，但鄰居擁有的片段若是該鄰居圈內的稀少片段，為維持稀少片段的最大存活率，則跳過此鄰居轉而挑選貢獻度第二低的鄰居，依此類推，如果最後所有鄰居都含有稀少的片段，則還是切斷貢獻度最低的鄰居的連結。

舉圖 3 為例，Peer A 在系統中遇到了下載瓶頸，因此挑出 B、C、D 這三個負載最大的鄰居，去各別找尋其鄰居(B1、B2、B3、C1、D1、D2)並建立候選人的初期清單，如圖所示。

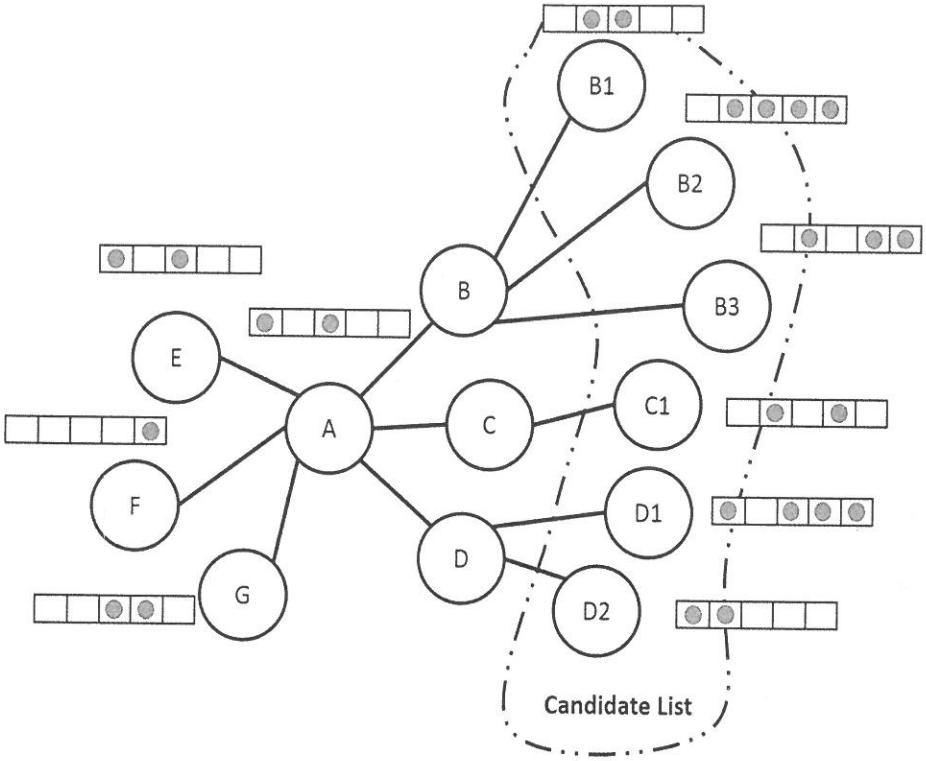

圖 3、初期候選人清單舉例

有了初期候選人清單後，Peer A 針對自己的頻寬及完成度去決定所要使用的策略，假設這些候選人的狀況如表 1 所示，節點 A 目前的狀態及更新鄰居策略可以分為下列三種情形:

表 1、鄰居候選人的現況

節點	頻寬	貢獻度	互補性
B1	高	3	4
B2	低	7	11
B3	高	7	12
C1	高	5	9
D1	低	4	5
D2	高	3	5

i. 若 Peer A 是個高頻寬且高完成度的節點(超過 40%的完成度門檻)，則 Peer A 會執行策略 1，自行變成一個幫助者，尋找低頻寬且低貢獻的節點來給予服務。雖然 B1 貢獻分數最小，但由於是高頻寬節點，不符合條件。而此處的 D1 節點能夠貢獻 A 的片段是 4 號片段和 5 號片段，加上缺少 4 號片段的鄰居(E、F)及缺少 5 號片段的鄰居(E、G)各有兩

個，計算出貢獻分數為 4，在候選清單中最小，因此 A 會去對 D1 作鄰居關係的建立。

ii. 現在假設完成度門檻設定為 60%，此處的 Peer A 是一個高頻寬、低完成度的節點，依照策略 2 的條件，它會去選擇互補分數最高的 C1 節點。其計算方法是，相互能夠給予的片段為 1 號、2 號、3 號、及 4 號片段，而 C1 可貢獻給 A 的片段是 2 號及 4 號，在 A 的鄰居圈也需要 2 號片段的有 E、F、G 節點三個，而也需要 4 號的有 E、F 兩個，因此 C1 對 Peer A 的貢獻分數等於 5。B3 雖然在候選清單的互補分數最大，但它已完成 60% 的片段下載，被視為高完成度的節點，不符合條件。而 B2 是低頻寬的鄰居，也不符合條件，因此 A 會選擇 C1 作為新鄰居。

iii. 最後我們假設 Peer A 是一個低頻寬的節點，它會遵循策略 3，並把自己視為一個尋求幫助的節點，進而尋找高頻寬且是高貢獻度的節點。從表 1 可以看出，B3 可貢獻給 A 的片段有 2 號、4 號及 5 號片段，而在 A 的鄰居圈內缺乏 2 號片段的節點有 3 個(E、F、G)，加上缺乏 4 號片段的節點有 2 個(E，F)，最後再加上缺乏 5 號片段的有 2 個(E、G)，總共為 7，且 B3 是高頻寬的節點，且貢獻分數最高，其分數為 7，因此 B3 最後會成為 A 的新鄰居。

接著，我們要舉例說明鄰居切斷連結的情況，圖 4 為鄰居連結以及其各個緩衝區的現狀:

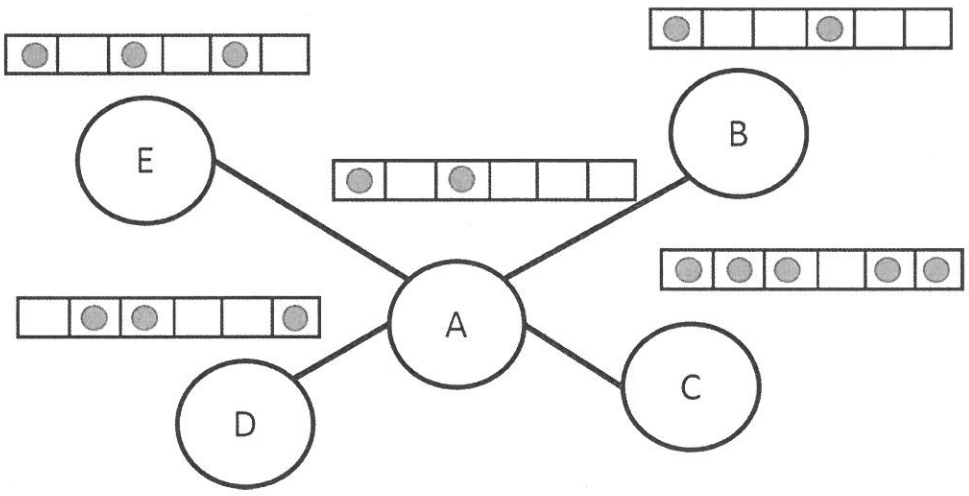

圖 4、切斷連結舉例

假設 Peer A 的鄰居數量已到達上限，無論是要去新增鄰居或是被加成新鄰居，都得先進行切斷鄰居的動作。我們將當下鄰居能夠給予 A 的片段貢獻量顯示於表 2，我們定義此處的稀少片段為在 A 鄰居圈內的鄰居是否擁有最稀少的片段，舉例來說，A 的鄰居 B 他擁有的 4 號片段，是其他鄰居都缺乏的，表示它包含稀少的 4 號片段。

表 2、欲切斷鄰居的現況

	貢獻片段	貢獻量	稀少片段
B	4 號	1	4 號
C	2 號、5 號，6 號	3	無
D	2 號，6 號	2	無
E	5 號	1	無

從表中得知每個鄰居可給予 A 的片段，加總形成的貢獻量。Peer B 和 Peer E 的貢獻量都是一個片段，但由於 B 含有在鄰居圈內屬於稀少片段的 4 號片段，因此保留其連結，而選擇去和 Peer E 作切斷鄰居關係。

我們的方法可以使得節點處於下載低效率的情況時，透過新增鄰居去變動其拓樸，利用不同的條件達到不同的目標。策略 1 的高頻寬節點找低頻寬節點和策略 3 的低頻寬節點找高頻寬節點會讓整體的拓樸漸漸變成高頻寬節點服務低頻寬節點的樹狀結構，如此在鄰居間的傳輸效益上會逐步提升。而利用策略 2 讓高頻寬的節點在早期可以與同樣為高頻寬的節點相互服務，快速擴張彼此間的片段來達到高完成度的門檻，進而執行策略 1，進行對於低頻寬節點的幫助。整體效能上由於逐步的加入了幫助與需被幫助的節點，使得最後平均完成時間表現會優於 BitTorrent 協定。

4. 效能評估

4.1 實驗環境

我們使用 Java 語言撰寫模擬程式來評估 SDTA 演算法的效能，實驗假設節點數量為 100 個

到 500 個不等，節點頻寬有高頻寬和低頻寬兩種，高頻寬節點的上傳頻寬為 6Mbps，下載頻寬為 10Mbps；低頻寬節點的上傳頻寬為 2Mbps，下載頻寬為 4Mbps。系統中存在一個 Seed，其上傳頻寬能力較大，為 24Mbps。每個節點的鄰居數量透過拓樸的更動，會從一開始的 10 個增加至最多 15 個。另外，為確保 SDTA 演算法並非只是因為增加鄰居數量而在效能上優於原始 BT，因此在原始 BT 的設置上，鄰居數量取 SDTA 的中間值 13 個。檔案大小設定為 600MB，分割成 300 個等份的 2MB 片段(piece)。低效能門檻及完成度門檻的設定以實驗結果來看，分別在 20%和 50%對整體效益是最好的，後面會再詳述原因。除 Seed 以外，我們假設一個節點完成檔案下載便會立即離開系統。數據如表 3 所列:

表 3、系統架構及實驗參數

Peer 數量	100~500
Peer 頻寬(高)	10Mbps 下載，6Mbps 上傳
Peer 頻寬(低)	4Mbps 下載，2Mbps 上傳
Seed 個數及頻寬	1 個，24Mbps 上傳
鄰居數量	SDTA:10~15，BT:13
Piece 數量及大小	300 個片段，每個 2MB
低效能門檻	20%
完成度門檻	50%

4.2 實驗結果

圖 5 呈現在不同的節點數量下平均下載時間的變化。我們可看出當節點數量增多時，平均下載時間也會隨之上升。SDTA 透過新增鄰居的機制，無論目標新鄰居貢獻度的高低，都能在及時提供片段的情況下，有效地減少下載時空等待的窘境，因此在整體平均下載時間皆會比拓樸維持固定的 BT 表現出 14%至 28%的改善。 在圖 6 中我們繪出了在線節點數量與在線節點之平均頻寬的關係。由於擁有較強能力的高頻寬節點，可以在較短時間內獲得較多片段，因此會較快

完成下載。由於節點在檔案下載完成後假設會立即離線，因此 BT 與 SDTA 在線節點的平均頻寬都會隨著高頻寬節點的離開而逐漸下降。然而 SDTA 的曲線會比 BT 來的趨緩，是因為在我們的策略中，高頻寬高完成度的節點會成為幫助者，幫助低頻寬的節點，此新增的低頻寬鄰居只能對他有低貢獻，因此高頻寬節點會再去找其他低頻寬的節點來幫助，漸漸會形成一個高頻寬的節點協助著許多低頻寬節點的狀態。我們的 SDTA 方法並非強制將高頻寬節點留到最後，而是讓高頻寬節點花較多的時間來服務低頻寬節點，但是高頻寬節點仍是會比低頻寬節點更早完成檔案下載。

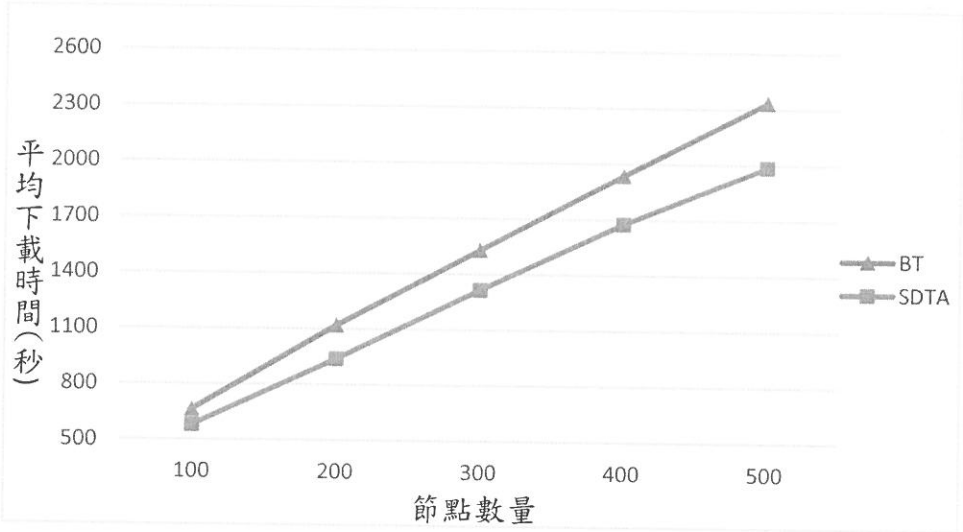

圖 5、平均下載時間與節點數量改變之關係

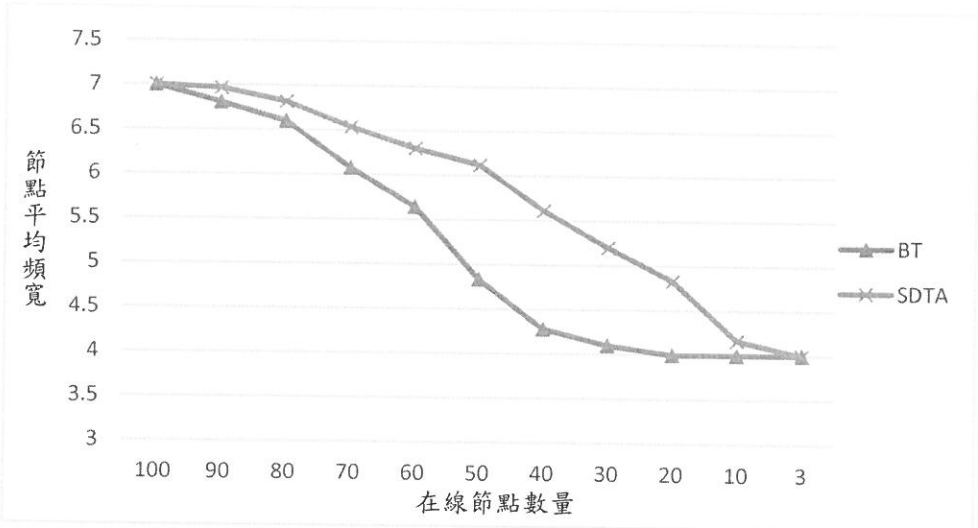

圖 6、在線節點的平均頻寬(100 個 peer)

圖 7 繪出高頻寬節點和低頻寬節點之間的比例改變對於平均完成時間之影響，SDTA 方法在比例為 6:4 時相對 BT 有著最佳的改善幅度，減少了 17%的下載時間。隨著低頻寬節點的比例增多，系統中服務能力較強的高頻寬節點減少，因此下載時間會逐漸上升。頻寬比例在 10:0 或是 0:10 的時候，所有節點無高低頻寬之分。我們觀察到雖然 SDTA 中的鄰居平均數量在前期會些許少於 BT，但隨著時間前進，下載完成的節點陸續離開系統後，BT 的鄰居平均數量會持續下降，而 SDTA 則是在下載中期及中後期仍有持續執行新增鄰居的動作，因此鄰居數量較 BT 為多，能夠在同時間內下載到更多片段的機率相對較大，所以 SDTA 的平均完成時間仍然能夠優於 BT 方法。

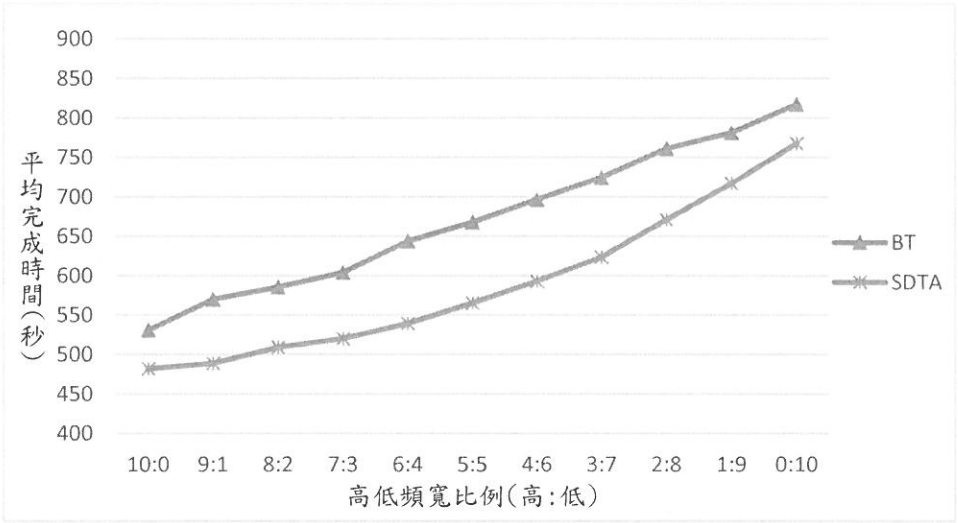

圖 7、系統高低頻寬數量之比例與平均下載時間比較(100 個 peer)

圖 8 呈現節點在不同低效能門檻之設置下，平均下載完成時間的變化。我們設定每 10 秒作一次低效能檢查，在此時間內，高低頻寬節點所能夠下載到的片段最多分別為 50 個及 20 個，在實際上難以達成所有節點都沒有下載延遲。因此門檻在 100%的情況下，鄰居增加率最大，致使達到鄰居數量上限後，處於得不斷更換鄰居，因而拓樸更動劇烈，造成許多已發送的片段請求因為連結被切斷而需要重新發送，因此等待時間變得更長。隨著門檻值的降低，能夠達到目標的節點數量增多，拓樸變動較趨和緩，使得平均下載時間跟著下降。由圖可觀察到在門檻值下降到 50%的時候，整體效益最好。當低效能的門檻小於 50%後，由於達成目標難易度減少，許

多節點可能在未發生低效能的情況下而未執行拓樸調整，漸漸偏向不作拓樸更動的 BT 效能，因此平均完成時間會再度上升。

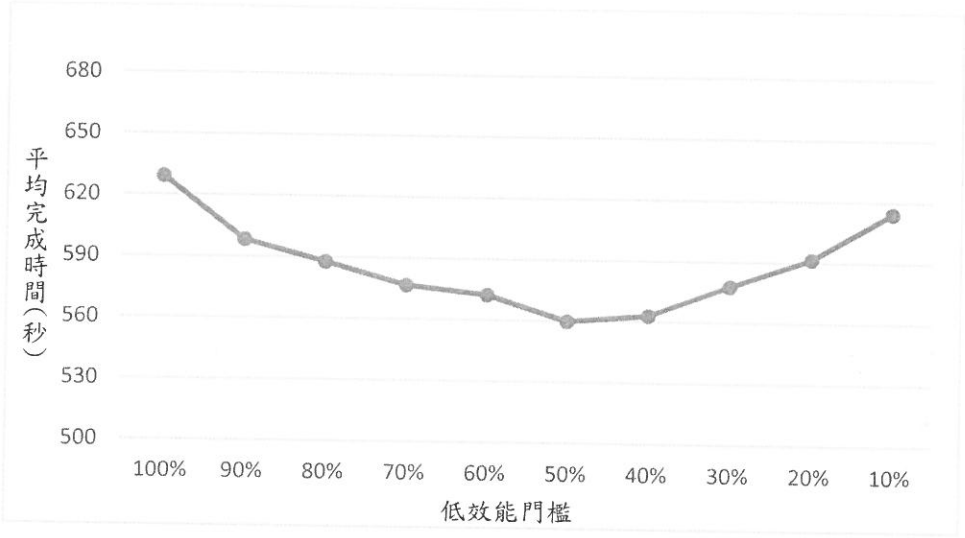

圖 8、低效能門檻與其平均完成時間之比較(100 個 peer)

圖 9 為完成度門檻值對於整體平均完成時間的比較。我們可觀察到同時發送出的片段要求數量越多，平均完成時間越小。發送片段要求的多寡之效益，差異在於高頻寬節點及 seed 的上傳能得到多大的應用。當每個節點最多只能有 1 個請求在外時，因為請求數量太少，大多數節點的頻寬處於閒置，當節點被允許發出 4 個請求的時候，幾乎可讓高頻寬的節點充份利用頻寬，達到不休息的狀態；若是增加到 8 個請求的時候，平均下載時間可以再降低。

由於 SDTA 演算法中只有高頻寬節點會依其完成度去選擇不同的策略，對於低頻寬節點較無差異，因此針對此圖，可把它視為高頻寬節點變成幫助者的時機點，落於何處會對於整體系統效能最好。由圖 9 可觀察出，完成度門檻於 20%的時候整體效益最佳。由於節點間的初始拓樸是隨機形成的，因此每個節點的高頻寬鄰居和低頻寬鄰居個數的平均比例為 1:1。當完成度門檻設為 10%的時候，假設一高頻寬節點 A 完成 10%的檔案下載，節點 A 成為幫助者的最佳情況就是其高頻寬的鄰居們能提供剩餘的 90%檔案片段。相反來說，若當下有 5 個高頻寬的鄰居，鄰居圈內無法完全補足節點 A 剩餘片段的機率為 $0.9^5 \approx 0.59$。同理可推，當門檻設為 20%的時候，高頻寬節點 A 完成自己 20%的檔案下載，其他高頻寬鄰居還未完成其餘 80%的片段下載之機率

約為 $0.8^5 \approx 0.32$。若節點 A 之前已因為是低完成度的 SDTA 策略而加入兩個高頻寬鄰居，則該機率降至約 $0.8^7 \approx 0.2$。此時鄰居內能夠提供節點 A 剩餘的檔案片段的高頻寬節點已足夠，因此 A 開始變成幫助者，去幫助低頻寬的節點。隨著完成度門檻的提高，高頻寬節點所需下載到的片段量增多，高頻寬鄰居之間的擁有片段覆蓋率增大，高頻寬鄰居能夠提供節點 A 尚未下載的片段之機率越來越低，此時越慢去幫助低頻寬的節點，造成下載完成時間之上升。

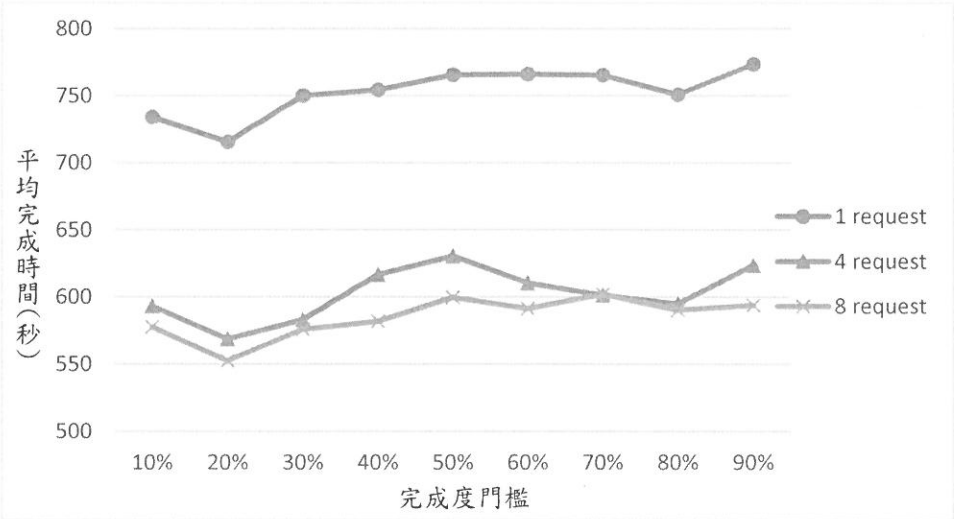

圖 9、完成度門檻與平均完成時間之關係(100 個 peer)

5. 結論

點對點技術的提出是透過充分利用每個使用者的上傳頻寬，有效降低伺服器的負擔。而且使用者越多，其傳輸頻寬總和越大，效率就越好，因此漸漸取代了主從式架構。在 BT 架構中，若節點的鄰居是固定的，當系統到達中後期時，許多鄰居因完成下載檔案而離開系統，在極端情況下在線節點只得和 Seed 為鄰居向其請求片段，即會變相形成主從式架構的傳輸方法，使得整體傳輸效率大打折扣。

為了改善上述問題，我們提出 Strategic and Dynamic Topology Adjustment (SDTA)演算法，當節點遇到下載瓶頸時，會依其完成度和頻寬去找尋相對應的目標新鄰居。找尋新鄰居的標準則

是基於其頻寬大小能力、檔案完成程度、對當下節點鄰居圈的貢獻程度、及對當下節點的互補程度，以這些條件綜合考量為依據，分為了三項策略。首先是策略二中，我們讓高頻寬低完成度的節點與其相同條件的節點作為新鄰居，透過高互補性達到互助之效果。且由於雙方皆為高頻寬，能夠快速讓雙方跨越高完成度門檻，然後變成幫助者執行策略一，去幫助低頻寬的節點。策略三則是將低頻寬節點直接設定成被幫助者，會不斷去尋求高頻寬鄰居的幫助。

模擬實驗結果顯示 SDTA 方法中，高頻寬節點在達到 20%的完成度門檻即成為幫助者時對整體效益最好，過早過晚都可能會有高頻寬節點平均下載被延宕，或是高頻寬節點大多離線，而使得低頻寬節點找不到能夠幫助的人的窘境。在總體平均完成時間來看，SDTA 方法相較原 BT 方法有著 13%至 28%的改善，這是因為拓樸的更動中還包含成為幫助者、盡快成為幫助者與尋求幫助者的幾項策略，讓 SDTA 能夠提供比 BT 更佳的效能。

參考文獻

[1] BitTorrent, http://www.bittorrent.com/

[2] N. Leibowitz, M. Ripeanu and A. Wierzbicki, "Deconstructing the Kazaa Network," *in 3rd IEEE Workshop on Internet Applications (WIAPP '03)*, Santa Clara CA, pp. 112-120, 2003.

[3] L. Sheng, X. Dong, J. Song and K. Xie, "Traffic Locality in the eMule System," *International Conference on Networking and Distributed Computing (ICNDC)*, pp. 387-391, October, 2010.

[4] PPTV, http://www.pptv.com/

[5] PPSteam, http://www.pps.tv/

[6] SKYPE, http://www.skype.com/zh-Hant/

[7] G. Tian, Z. Duan, T. Baumeister and Y. Dong , "A Traceback Attack on Freenet," *IEEE INFOCOM,* pp. 1797-1805, April 2013.

[8] F. Dabek, E. Brunskill, M. F. Kaashoek, D. Karger R. Morris, I. Stoica and H. Balakrishnan, "Building Peer-to-Peer Systems With Chord, a Distributed Lookup Service," *Proceedings of the 8th Workshop on Hot Topics in Operating Systems*, 2001.

[9] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," *Lecture Notes in Computer Science*, vol. 2218, pp. 329-350,

2001.

[10] H. Nakano, K. Harumoto and S. Nishio, "Topology Re-formation Algorithms for Ubiquitous P2P Networks Based on Response Statistics," *the 2007 International Symposium on Applications and the Internet Workshops*, 2007.

[11] P. Slivey and L. Hurwitz, "Adapting Peer-to-Peer Topologies to Improve System Performance," *the 37th Hawaii International Conference on System Sciences,* 2004.

[12] Z. Zhou, Z. Li and G. Xie, "ACNS: Adaptive Complementary Neighbor Selection in BitTorrent-like Applications," *IEEE International Conference on Communications (ICC)*, 2009.

[13] 郭成霖、呂俊賢、陳彥宇，「利用延遲高頻寬節點之下載完成度以改善 BitTorrent 類型系統之檔案下載」，2014 全國電信研討會，2014 年 11 月，台灣。