國立交通大學

網路工程研究所

碩 士 論 文

針對在不同種類的同儕之中增強資料散播的一種 BitTorrent 修改方法

A BitTorrent Modification for Enhancing Data Dissemination among Disparate Peers

研 究 生：李明龍

指導教授：邵家健　教授

中 華 民 國　九 十 六　年 九 月

針對在不同種類的同儕之中增強資料散播的一種BitTorrent修改方法

學生：李明龍　　指導教授：邵家健

國立交通大學網路工程研究所 碩士班

摘　　要

我們的目的是想要解決在 BitTorrent 協議內並未被重視但卻有可能發生的兩個問題：第一，過於稀有的檔案資訊的散播在 BitTorrent 協議內是非常重要的問題，若是沒有將這些過於稀有的資訊散發出去會使得檔案片段分布狀態並不是非常平均，因此可能會使得整體的下載速度變慢；另外，過於集中的檔案資訊的散播也是擁有相同的問題。第二，無法禁止同儕在完成檔案下載時就離開分享的行列，這樣一來上傳頻寬高的同儕如果擁有那些過於稀有的檔案資訊或是過於集中的檔案資訊的同儕在下載完所有檔案片段就不再分享，則剩下的那些同儕就有可能無法下載完整的檔案。

我們提出兩種方法來解決上述的問題。首先，我們將下載階段分成三階段，每一個階段使用不同的上傳同儕選擇演算法，如此可以有效針對過於集中的資訊的散播，且要求大家必須多分享一些時間才能離開。接著，若是 BT 客戶端發現有過於稀有的資訊，則會盡全力去向擁有這些資訊的同儕要求拿取。

從實驗中我們觀察到幾項結果並得到兩個結論：一、資料的散播速度的確比原本的 BitTorrent 快；二、整體下載速度大幅增快。這些結果符合我們原本的期望，證明我們的方法表現的比原本的 BitTorrent 好。

A BitTorrent Modification for Enhancing Data Dissemination among Disparate Peers

Student: Ming-Lung Li Advisor: John Kar-Kin Zao

Institute of Network Engineering
National Chiao Tung University

ABSTRACT

Our purpose is to resolve two problems which are important but always ignored in BitTorrent. First, it is urgent to distribute extremely scarce pieces. If these pieces do not be broadcasted, these pieces will not be spread very well and decrease download speed of peers in the swarm. Otherwise, the seriousness of dispersing concentrated pieces is the same as scarce pieces. Second, BitTorrent client cannot request peers who download completed to stay in the swarm and share with other peers. Peers with high upload rates always download completed quicker than slow ones. If they have scarce pieces or concentrated pieces and close BitTorrent client after finishing download, the other peers in downloading group will not download completed file.

We provide two methods to settle previous problems. First, we divide three stages when the BitTorrent client is downloading. Our revised BitTorrent client use different unchoke algorithm in each stage. It will help owner of concentrated pieces distribute those pieces and prolong on-line duration of peers. If revised BitTorrent client find the scarce pieces, it will do its utmost to request the scarce pieces.

We observe several results from the experiments and get 2 conclusions. (1) the speed of spreading data of revised BitTorrent is faster than the original one; (2) overall average download speed of revised BitTorrent is far quicker than the original one. The results fulfill our expectations. The performance of revised BitTorrent is really better than the original one.

誌謝

感謝邵家健老師這兩年多的殷勤指導，在與老師討論的過程中學習到非常多東西，並且在老師循循善誘的教導之下，我對於做研究終於了解一點皮毛，企盼以後能夠將所學貢獻於社會。感謝張哲維學長幫助我釐清許多研究上的疑惑，讓我可以不致於走入死胡同。更是感謝郭芳伯，沒有他跟我一起討論，我也會一直卡在依些小細節上，並且幫助我設計和撰寫分析紀錄檔的程式，讓我可以那麼快就將所有資料清楚完整的表達。感謝黃國晉學長幫忙我抒發一些研究不順利的心情。感謝朱書玄學長來實驗室都會請我吃東西。也感謝其他實驗室的夥伴：輔國、哲民、育志、凌軒的支持與照顧。最後要感謝彥君在我人生中最低潮的時候給我打氣，謝謝你！

目錄

圖表目錄

表格目錄

Chapter 1 Introduction

由於近年網路技術的蓬勃發展，家用網路速度因而不斷向上提升，使得許多網路使用者想要藉由某些軟體來分享自己喜歡的影片或是自己製作的小遊戲或軟體，為了因應此需求，Bram Cohen 製作了 *BitTorrent*[1][2]這套軟體，利用同儕計算(Peer-to-Peer)的方式在網路中許多使用者之間分享檔案。我的研究便是針對 *BitTorrent* 架構做一些修改，使其能夠更為輕鬆地處理某些情況。

1.1 Problem Statement

1. 我們想要解決當利用 *BitTorrent* 下載檔案時，若發現其中某些檔案片段過於少人擁有，必須找出一套方法盡快將這些過於稀有的檔案片段發送出去。
2. 我們想要解決當利用 *BitTorrent* 下載檔案時，若發現其中某些檔案只被某一特定族群的同儕擁有，必須找出一套方法盡快將這些過於集中的檔案片段發送出去。
3. 利用一些小方法延長同儕分享的時間，以免使用者一下載完檔案就急著離開分享的行列，造成某些檔案片段無人擁有，因而使得仍在下載中的同儕無法完全下載檔案。

1.2 Research Approach

在這篇論文中，我們提出一套新的加權函數(weight function)應用於選擇上傳同儕(upload peer)時，此函數內含許多我們對於之前所提出的問題有相關的因素(element)，並且定義許多在不同情形下相對應的比重權值(weight factor)，選出最適合的上傳同儕，以期我們可以對於這些問題提出令人滿意的解答。另外我們根據下載檔案的完成度分成數個階段，每個階段都有不同的任務，因此每個階段的

比重權值也不盡相同。最後為了找出更多擁有過於稀有的檔案片段的同儕，我們設計了一套比重權值套入加權函數找出對我們而言最沒有益處的同儕並踢除，使得我們可以跟更多其他同儕建立連線，進而找出擁有過於稀有檔案片段的同儕。

1.3 Outline of Thesis

剩餘的論文章節組織如下：第二章，我們會介紹關於 *BitTorrent* 的背景知識以及其他相關研究的介紹；第三章，我們會詳細介紹所提出的方法和機制；第四章，我們會展示出模擬的成果並評估我們的方法；第五章，我們對於此論文做總結並提出未來的工作方向。

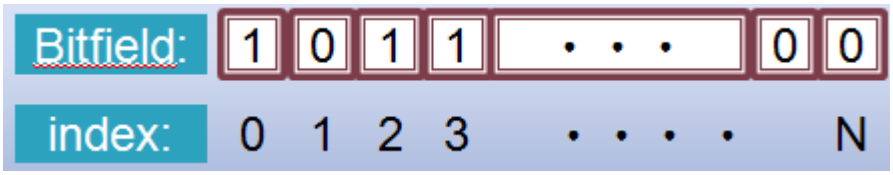
Chapter 2 Background

在此章中我們會簡短介紹 *BitTorrent* 的相關知識以及與我們題目相關的一些研究。

2.1 *BitTorrent*

BitTorrent 是一套同儕計算架構的檔案分享軟體，擁有原始檔案的使用者利用 *BitTorrent* 客戶端(Client，以下簡稱 BT 客戶端)製作 torrent 檔案，這些原始檔案可以是影片、音樂、文件或是安裝程式等，想要下載檔案的使用者必須抓取那些在各大網站發佈的 torrent 檔案，並利用 BT 客戶端開啟 torrent 檔案即可準備開始下載檔案。BT 客戶端開啟 torrent 檔案之後會知道同儕列表記錄者(tracker)所在，同儕列表紀錄者主要是負責記錄目前有哪些同儕正在分享或下載此檔案，每個同儕都會定期與它連線並更新自身最新的情況，因此與其建立連線後可取得同時正在分享此檔案的同儕列表(peer list)，接著與這些在同儕列表上的同儕們建立連線，連線建立成功後會與同儕互相交換內容簡表(bit_field)，內容簡表裡面記錄著同儕對於各個檔案片段的擁有與否。這邊我們先介紹檔案片段，在製作 torrent 檔案的時候，BT 客戶端會將原始檔案切割成數個固定大小的檔案片段(piece)，除了最後一個檔案片段不一定是相同大小。所以我們可以把內容簡表想像成一個陣列如圖表一，舉例來說 bit_field[0]的值是 1，表示這個同儕擁有第一個檔案片段；bit_field[1]的值是 0，表示這個同儕現在不完全擁有第二個檔案片段。互相交換 bit_field 代表 BT 客戶端與同儕互相知道對方現在擁有哪些檔案片段。同儕與每個已建立連線的同儕互相都有兩個參數來掌控他們之間的行為，像是下載東西或是上傳東西，而這兩個參數分別是 interest 和 choke，BT 客戶端與連線中的每個同儕各自擁有一份，在 BT 客戶端中 interest 代表 BT 客戶端是否對於同儕擁有的檔案片段感興趣，而 choke 代表 BT 客戶端是否禁止同儕從這邊下載東西，

舉例來說，當 BT 客戶端的 interest=1、choke=0 時，代表它對這個同儕擁有的檔案片段有興趣下載且允許這個同儕從它這邊下載東西。所以同儕與同儕之間必須小心維護對方以及自身的兩個參數狀態，我們可以說這兩個參數是整個核心所在，尤其是如何選擇上傳同儕更是關鍵。



圖表一：bit_field 想像圖

現行上傳同儕選擇演算法(unchoke algorithm)考慮的因素有幾個，第一先將過去 20 秒內上傳資料給 BT 客戶端速度大於 256 bytes/sec 的那幾個同儕根據其上傳速度由快至慢做排序並依序開放它們從 BT 客戶端下載東西，我們稱此種行為模式為以牙還牙(tit-for-tat)，由於每個同儕可以上傳(unchoke，在這裡代表 choke=0)給其他同儕的個數不同[1]，所以依序將同儕的 choke 設為 0 直到沒有剩餘空位；若是還有其他空位可以上傳資料給其他同儕，先檢查上傳給同儕的資料量與從同儕那邊下載的資料量比例是否大於 3，超過則不考慮開放此同儕從 BT 客戶端這邊下載東西，反之將比例小於 3 的同儕依照 BT 客戶端從他們那邊各自下載的資料量由大至小做排序並依序開放讓他們可以從 BT 客戶端下載資料。另外為了讓那些剛開始下載檔案而尚未擁有完整的檔案片段的同儕可以順利得到幾個完整的檔案片段，使它可以有籌碼跟其它同儕交換，且為了讓 BT 客戶端有機會找到其他條件更好的同儕，所以設計了樂觀選擇上傳同儕(optimistic unchoke)這個方法。由於樂觀選擇上傳同儕主要是為了搶救那些較為弱勢的同儕，因此它考慮的因素會比較照顧這些同儕，每隔 30 秒 Azureus[3]客戶端會利用上傳資料量減掉下載資料量的差值做排序，差值越大則越有機會被選為樂觀選擇上傳的同儕，因為越剛開始下載的同儕越沒有東西跟別人分享。

[1] 這是依照同儕自身的上傳速率決定，上傳速率越高則可以開放越多其他同儕從 BT 客戶端這邊下載東西

現在若是有個已連線的同儕允許 BT 客戶端從他那邊下載東西，它會選擇最少人擁有且還沒下載的那塊檔案片段，但不是一次要求一整塊檔案片段，因為檔案片段還會被分割成數個固定大小的檔案區塊(block)，檔案區塊才是最小的資料交換單位，所以 BT 客戶端只能一次要求數個檔案區塊。

2.2 Related Work

接下來我們會簡單介紹一些有關改善 *BitTorrent* 的研究。

2.2.1 Analyzing and Improving *BitTorrent* Performance

此篇文章[4]利用模擬的方式解讀 *BitTorrent*，模擬的結果顯示現在 *BitTorrent* 的機制可以使得同儕的上傳速率被有效的利用，且在一般情形下檔案下載時間也非常快速。文章作者提出一個新的方法來改善現在 *BitTorrent* 可能有不公平的地方，他提出若要開放某個同儕從 BT 客戶端這邊下載，則先必須符合下列這個算式

$$U_{ab} \le D_{ab} + \Delta \qquad (1)$$

其中U_{ab}代表同儕 A 上傳給同儕 B 的檔案區塊數、D_{ab}代表同儕 A 自同儕 B 下載的檔案區塊數而Δ代表在這個連線中所可以忍受的不公平門檻(threshold)區塊數，因此每個同儕最多必須多送的不公平區塊數目就是$d\Delta$，d 代表 BT 客戶端與多少同儕建立連線的數目，這樣一來就可以確實防止想要投機取巧不想上傳卻想下載檔案的使用者，但卻也付出了一些代價，像是沒有好好利用上傳頻寬。另外為了解決上傳同儕發送資料給下傳同儕時，因為頻寬不對等的關係造成上傳頻寬浪費或是下載頻寬閒置的情況，所以他們提出修改 tracker 使得它可以按照同儕報告的頻寬來挑選差不多頻寬的同儕並回傳給他們，不過同儕在跟 tracker 溝通之時必須回報 tracker 它的頻寬大小，結果真的有大幅改善原本的情況，但須信任 BT 客戶端回報的頻寬是正確的。

2.2.2 Rare First and Choke Algorithms are Enough

此篇文章[5]使用 *mainline*[2]的 BT 客戶端實際執行 torrent 檔案並且於客戶端內紀錄相關資訊證明現在的兩個策略—稀有的檔案片段最先被要求以及選擇上傳同儕演算法—是非常公平且非常健全。但此篇文章提出了幾個有趣的問題：第一，在一開始分享檔案時，通常只有來源提供者擁有完整的檔案，因此在這種情形下來源提供者必須盡快散播它自身擁有的檔案，使得每塊檔案片段可以有許多備份，所以來源提供者離開分享行列時可以不用擔心仍然有些檔案片段非常稀有；第二，當一個新的同儕想要下載檔案時，它拿到第一塊檔案區塊的時間是否可以再縮短一些，也就是讓新加入的同儕可以快點拿到幾塊完整的檔案片段，因而有籌碼跟其他同儕交換。

原本做種者(seeder[3])選擇開放同儕下載資料的方法是將那些從做種者這邊下載的同儕依照下載速度由快至慢排序並依序上傳資料給它們。但作者提出新的上傳同儕選擇演算法，他們對於做種者選擇上傳同儕的方法改成每 30 秒隨機選擇一個新的同儕並開放它下載，將同儕再根據被開放的時間由近至遠排序，禁止第 4 個之後的同儕下載。如此一來每個跟做種者連線的同儕都可以在一定的時間內被開放從做種者那邊下載東西。我們覺得這種方法並不能最快的將做種者擁有的資訊以最快的速度複製一份到分享族群中，因為不是每個同儕的下載速度都很快，若是遇到下載速度較慢的同儕，雖然做種者的上傳速率很高，但被選到的上傳同儕並無法快速的收下並散播出去，因此有可能拖慢資訊的散播速度。

2.2.3 Missing Piece Issue and Upload Strategies

此篇文章[6]提出新的檔案片段選擇策略，現在的 BT 客戶端是由被開放下載的同儕要求想要的檔案片段，但作者卻想要改變成由開放下載的同儕選擇他要上傳哪

---

[2] *mainline* 是由 Bram Cohen 自行修改的 BT 客戶端。

[3]指得是只分享檔案而不下載檔案的同儕。

些資訊給哪一個同儕。方式改變之後便有兩種方法可以選擇上傳的同儕(i)跟上傳的檔案片段(j)，我們在此簡寫為(i, j)，第一種是把全部有可能的配對都列出來，再利用隨機選擇要上傳的同儕跟檔案片段；第二種方法是利用一些選擇策略從要上傳的檔案片段或是要上傳的同儕選擇其一，像是稀有的檔案片段優先或是擁有比較少的同儕優先，這兩種策略僅能選擇一種，第一項選擇完之後再來選擇相對應的第二項。此篇文章僅提出想法並未實際產生數據證明可於現實網路中表現良好。

2.2.4 Improve the Download Time of BitTorrent-like Systems

此篇文章[7]提出一個新的方法選擇要下載的檔案片段，BT 客戶端根據每個同儕的內容簡表，查出有哪些尚未下載且其它同儕也需要的檔案片段，並計算每塊檔案片段的需求程度。每個同儕對於缺少的檔案片段的需求程度與其本身的已下載檔案的完成度成正比，同儕的完成度越高則比重權值越大，將每個同儕對於每塊檔案片段的比重權值加總，找出得分最高的檔案片段並下載。我們感覺此方法只是原本的 BitTorrent 同儕選擇先拿稀少的檔案片段(Rarest First)的變種，但的確可以減少一點整體的平均下載時間。

Chapter 3 Concepts of Research

3.1 Principles and Objectives

本研究中所設計的演算法和機制皆基於以下幾點目標與原則：

1. 使用簡單的加權函數以及比重權值，讓 BT 客戶端可以迅速決定須接受動作[4]的同儕。
2. 在下載初期，我們令 BT 客戶端的行為以下載為優先；在下載中期時，令 BT 客戶端的行為模式與原本的行為模式相同，必須用自身的上傳頻寬去換得下載的機會；在下載晚期，我們令 BT 客戶端的行為以分享為優先。
3. 在某些檔案片段只被極少數的同儕擁有時，那些尚未下載並發現此情形的同儕必須竭盡全力取得下載稀有檔案片段或是過於集中的檔案片段的機會。

接下來我們將對於之前所介紹的問題提出解決的演算法與機制。在 3.2 節中，我們會介紹對於過於稀少的檔案片段處理的方法；在 3.3 節中，我們會介紹對於過於集中的檔案片段處理的方法以及介紹如何延長 *BitTorrent* 使用者分享的時間。

3.2 Deal with Extremely Rare Pieces

BT 客戶端對於每個檔案片段都有紀錄總共幾個同儕擁有，為了定義哪些檔案片段是屬於過於少人擁有，因此我們設置了兩個參數—低水線及高水線，若是檔案片段擁有人數低於低水線，我們稱此檔案片段為過於稀有的檔案片段。當我們修改過的 BT 客戶端發現其中有些檔案片段是過於稀有的且自身尚未擁有，則會進入緊急狀況，此時 BT 客戶端會開放給那些擁有過於稀有的檔案片段的同儕下載，主要是為了取得那些同儕也開放給 BT 客戶端下載的機會，讓 BT 客戶端可以下載那些過於稀有的檔案片段，因為上傳同儕選擇演算法大部分的時間還是考

[4] 被開放下載或是被捨棄

慮以牙還牙的精神，也就是某同儕開放給 BT 客戶端下載越快越多，BT 客戶端也會盡全力給對它最好的同儕開放下載。

另一方面，若是 BT 客戶端所允許的最大連線數已經被佔滿，為了找到其他擁有過於稀有檔案片段的同儕，BT 客戶端必須要捨棄掉某些同儕，在此我們採取的方法是首先踢掉那些驕傲的同儕[5]，若是還需要踢掉其他同儕，則再按照同儕開放 BT 客戶端下載的時間的總和以及同儕的上傳速率來排序，如果同儕上傳給我速率越快且選擇我為上傳同儕的時間越多，則 BT 客戶端越不會踢掉他們；反之，若是同儕上傳給我速率越慢或是常常不選擇我為上傳同儕，BT 客戶端有越大的機會選擇將這些同儕踢掉。但是如果被選中的同儕擁有低於高水線的檔案片段，我們必須跳過此同儕而考慮下一個同儕。保護擁有低於低水線的檔案片段的同儕原因很明顯是因為不想這些同儕棄我而去，使得使用者無法得到完整的檔案；但保護擁有介於高水線與低水線之間的檔案片段的同儕的原因是雖然某個檔案片段的擁有人數高於低水線，但是此檔案片段擁有的人數可能只是略高於低水線所設定的人數，若是剛好連續與數個擁有此檔案片段的同儕中斷連線則會導致此檔案片段成為過於稀有的檔案片段，情況因此又會變得更糟，所以我們設定保護那些擁有人數低於高水線的檔案片段。

3.3 Deal with Concentrated Pieces

擁有過於集中檔案片段的意思指得是雖然 BT 客戶端與 50 個同儕建立連線，但是卻不一定跟每個同儕交換資料，根據[8]表示如果沒有任何一個做種者分享檔案，上傳速度快的同儕總是與上傳速度快的同儕交換檔案，因此可能會導致某些檔案片段總是在上傳速度快的同儕手中，而上傳速度慢的同儕無法與那些上傳速度快的同儕交換檔案，且若是這些上傳速度較快的同儕一下載完就離開分享的行列，有些同儕就無法獲得完整的檔案。

[5] 驕傲的同儕指的是那些雖然它開放讓 BT 客戶端下載，但是當 BT 客戶端請求同儕傳送想要的檔案區塊時卻被忽略，而不送給 BT 客戶端要求的檔案區塊。

為了解決這個問題，我們提出了將上傳同儕選擇演算法依照完成度的不同採用不同的選擇上傳同儕的原則。我們將下載的完成度分成三個階段，分別是低於30%、介於30%和70%之間以及大於70%。以下是介紹我們對於這三個不同階段分別設計的選擇上傳同儕的原則：

- 下載完成度低於30%的選擇上傳同儕原則：第一是開放那些上傳給我速度最快的同儕；第二是開放給同樣是完成度低於30%的同儕下載，且擁有的內容跟BT客戶端所擁有的差異越大就越有機會。BT客戶端將每個同儕的資訊套入下列函式計算

  $f_i = 2 \times r_i + \lceil 0.3 - c_i \rceil \times d_i, i \in C$

  r_i：同儕 i 上傳給我的速率除以最快的同儕上傳速率。

  c_i：同儕 i 的下載完成度。

  d_i：BT客戶端擁有那些同儕 i 所沒有的檔案片段的數量除以BT客戶端總共擁有的檔案片段的數量。

  f_i：同儕 i 經過計算得到的分數。

  C：與BT客戶端建立連線的同儕集合(peer set)。

  接著BT客戶端將每個同儕的 f_i 由高至低排序，選出分數最高的四個同儕並開放讓它們下載。

- 完成度介於30%和70%之間的選擇上傳同儕原則：第一是使用原本*BitTorrent*的上傳同儕選擇演算法；第二是開放一個上傳的位置無償讓完成度低於30%的同儕輪流下載。BT客戶端使用以下函式計算每個同儕的分數

  $f_i = \lceil 0.3 - c_i \rceil \times (t_N - t_i), i \in C$

  t_i：我上次傳送有效資料給同儕 i 的時間。

  t_N：現在的時間。

  c_i：同儕 i 的下載完成度。

  f_i：同儕 i 經過計算得到的分數。

  C：與BT客戶端建立連線的同儕集合(peer set)。

接著 BT 客戶端將每個同儕的 f_i 由高至低排序，選出分數最高的同儕並開放讓它下載。

- 完成度大於 70%的上傳同儕選擇原則：第一，盡量延長 BT 客戶端分享的時間；第二，對完成度 30%以下的同儕多加照顧，使得這些同儕可以在很短的時間內與別的同儕分享資料並藉此拉高下載速度。我們的做法是先將與 BT 客戶端建立連線的同儕集合隨機排序成一張表，接著 BT 客戶端按照順序開放給表上對應的同儕下載，且按照同儕的完成度來決定開放下載的時間，對於完成度低於 30%的同儕會開放 90 秒的時間讓它們下載；對於介於完成度 30%與 70%之間的同儕會開放 30 秒的時間讓它們下載；最後，對於完成度大於 70%的同儕會開放 10 秒鐘讓它們下載。

對於完成度為 100%的 unchoke 原則我們設為與原本 BitTorrent 的原則相同，就是對那些從 BT 客戶端下載速度最快的同儕較好，接收速度越快越有機會被我開放下載，這樣才能以最快的速度散播資訊。

Chapter 4 Research Result

在此章節我們將會顯示我們演算法表現的結果，並就這些圖表來探討我們所設計的演算法與原本 *BitTorrent* 的演算法其中的相異處。

4.1 System Configuration

我們在此論文中使用 Azureus 做為我們修改的 *BitTorrent* 客戶端平臺，這是一個被廣為使用的 *BitTorrent* 客戶端，而且程式碼是以 Java 撰寫，因此能夠在不同的作業平台上運作，並且提供同儕列表記錄者的功能，讓我們可以在實驗期間輕鬆的不受外來的同儕影響建立一個乾淨的分享環境。

我們使用分佈在不同地方的 21 臺電腦執行我們所修改的 BT*[6]來進行實驗，關於電腦的分布地點以及網路頻寬如表格 1 所示。

[6] 這邊我們稱修改過的 Azureus BitTorrent client 為 BT*，原本的 Azureus BitTorrent client 為 BT。

表格 1：實驗用電腦詳細資料

Peer Number	Position	Network Type
1	Hsinchu	FTTB (10Mb/2Mb)
2	Hsinchu	TANet
3	Taoyuan	TANet
4	Hsinchu	FTTB (10Mb/2Mb)
5	Hsinchu	TANet
6	Yunlin	TANet
7	Taipei	ADSL (8Mb/1Mb)
8	Taipei	ADSL (8Mb/640Kb)
9	Hsinchu	ADSL (4Mb/1Mb)
10	Taipei	ADSL (8Mb/2Mb)
11	Hsinchu	ADSL (8Mb/640Kb)
12	Tainan	ADSL (8Mb/640Kb)
13	Sinjhuang	ADSL (8Mb/640Kb)
14	Sinjhuang	ADSL (2Mb/256Kb)
15	Taipei	ADSL (2Mb/256Kb)
16	Taipei	ADSL (2Mb/256Kb)
17	Hsinchu	ADSL (8Mb/640Kb)
18	Sinjhuang	ADSL (2Mb/256Kb)
19	Hsinchu	ADSL (2Mb/256Kb)
20	Miaoli	ADSL (2Mb/256Kb)
21	Hsinchu	TANet

我們的實驗使用編號 21 的電腦當作每次的做種者，並且設定其上傳速率為 110KBps。編號 1 到 20 的電腦就當成是欲抓取檔案的下載者，由於不是每次都可以使用 20 台電腦做為下載者，因此大部分的數據都是由少於 20 個下載者所產生。我們在實驗中使用下列的設定：

- 所使用的分享測試檔案大小為 260.9MBytes，Azureus 將此檔案產生一新.torrent 檔並將測試檔案分割為 1044 個檔案片段，每塊檔案片段為 256KBytes。
- 每個實驗從頭到尾都只有一個做種者，我們計算數據時並未將該做種者的資料列入考量。

- 每個下載者都是在同一時間開始下載檔案，一下載完檔案就馬上離開分享群體。
- 每個下載者的同時可上傳的人數上限是由 Azureus 利用該下載者的上傳頻寬來決定。

在每個實驗中，我們準備測量許多數據來證明我們所提出的架構的表現的確優於原本的 BitTorrent 架構，而這些數據分別是平均的資料散播速度、平均每個同儕的資料累積速度及每個同儕下載檔案個別花費的時間。

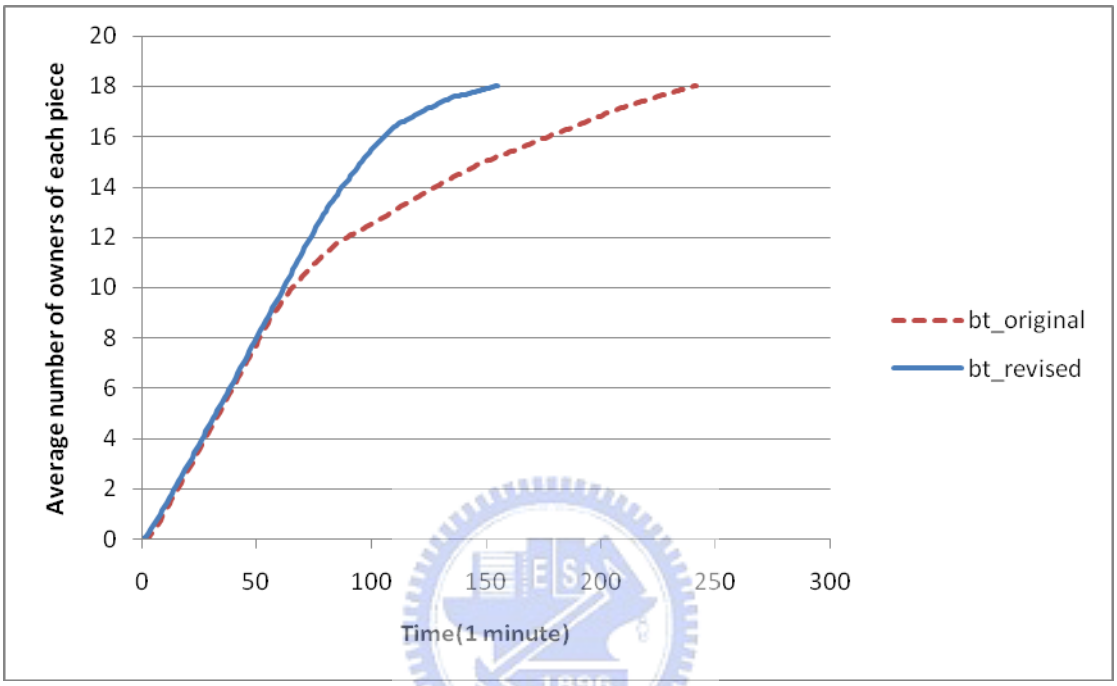
4.2 Results and Analysis

我們做了許多次實驗，每次的檔案下載者人數都不大相同，因為每次實驗所有現象都大同小異，所以我們抽三組出來放在論文中，其中我們使用一組來說明我們修改的 BT*的表現的確是比原本的 BT 好，其他兩組我們放在附錄中，讓有興趣的人可以自行比較。在此節中，我們用的圖由 18 個檔案下載者的紀錄統計得出，這 18 個檔案下載者依照上傳頻寬分成 6 組，此 6 組的頻寬分別是 80KBps[7]、60KBps、40KBps、20KBps、10KBps 和 5KBps，我們按照每台電腦的網路頻寬分別分到適當的組別，且每組各自擁有 3 台電腦。

- **資料的散播速度：**如圖表二所示，圖表二所代表的意義是在每個時間點每塊資料片段平均被多少下載者擁有，我們可以發現一開始原本的 BT 與我們所修改的 BT*的散播資料的速度差不多，到了下載中期之後，原本的 BT 開始降低資料的散播速度，而 BT*卻持續以相同的速度繼續散播資料，直到下載末期 BT*才降低散播資料的速度。圖表四所代表的意義為在每個時間點每塊檔案片段的擁有人數的標準差，我們可以看到一開始 BT*的曲線上升的比原本的 BT 快上許多且最頂端的值也比原本的 BT 大，這代表的是 BT*資料散播的速度相當快，當一個同儕拿到了新的檔案片段，其餘的同儕無不渴望拿

[7] KBps = Kilo Bytes per Second

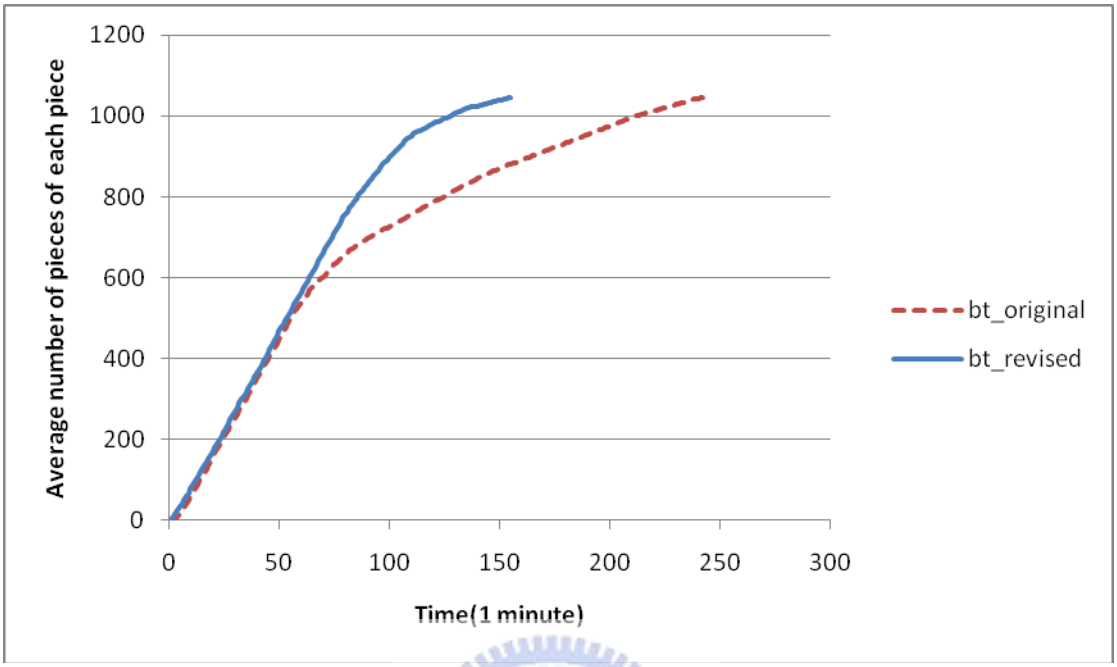
到這個稀有的檔案片段，於是造成每個檔案片段的擁有人數的差異性如此之大；一但過了峰頂之後，BT*曲線的下降速度也是比原本的 BT 來得快，因為同儕所擁有的檔案片段數越來越多，加上 BT*的散播速度又比 BT 快，所以造成這個現象。



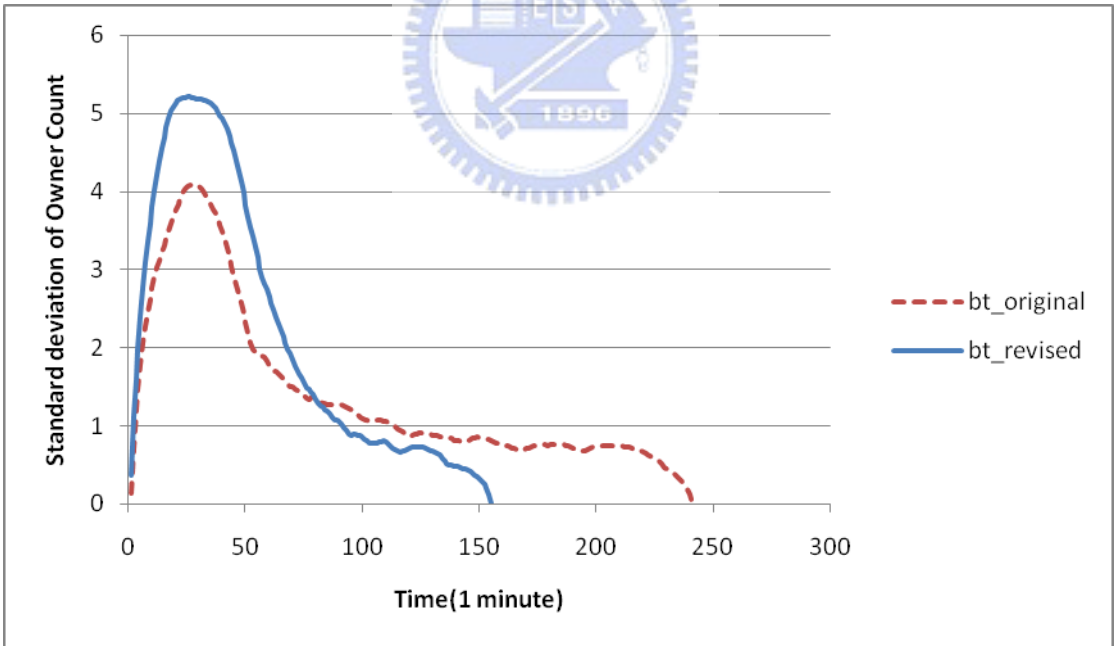
圖表二：Data Diffusion (Avg. Peer Count / Piece)

- **資料的累積速度：**如圖表三所示，此圖所代表的意義為在每個時間點每個下載者平均擁有多少塊檔案片段，我們可以輕易地發現在下載中期之後，BT*平均每個下載者拿取資料的速度明顯優於原本的 BT。
- **結論：**看過圖表二、圖表三和圖表四之後，我們可以得到一個結論，現行的 BT 雖然使用非常簡單的上傳同儕選擇演算法，但是在一開始的表現卻與我們比較聰明的 BT*大致相同，不過由於在下載中期之後速度較快的同儕紛紛退出分享行列，某些檔案片段的擁有人數因此減少，使得某些檔案片段的擁有人數掉到低水線以下，讓原本的 BT 分享狀態變得不穩定，造成平均的資料下載速度減緩；反觀我們設計的 BT*並不會因為速度快的同儕的退出造成整體資料下載速度減緩，這是因為雖然這些速度快的同儕退出分享群體，但是每塊檔案片段的擁有人數卻保持固定在高水線以上，直到下載末期之後才

掉回到高水線以下，但是仍然保持在低水線以上，到下載的最末段的時間才掉回低水線以下，不過這也是因為當時的下載者人數已經低於低水線了。



圖表三：Data Accumulation (Avg. Piece Count / Peer)

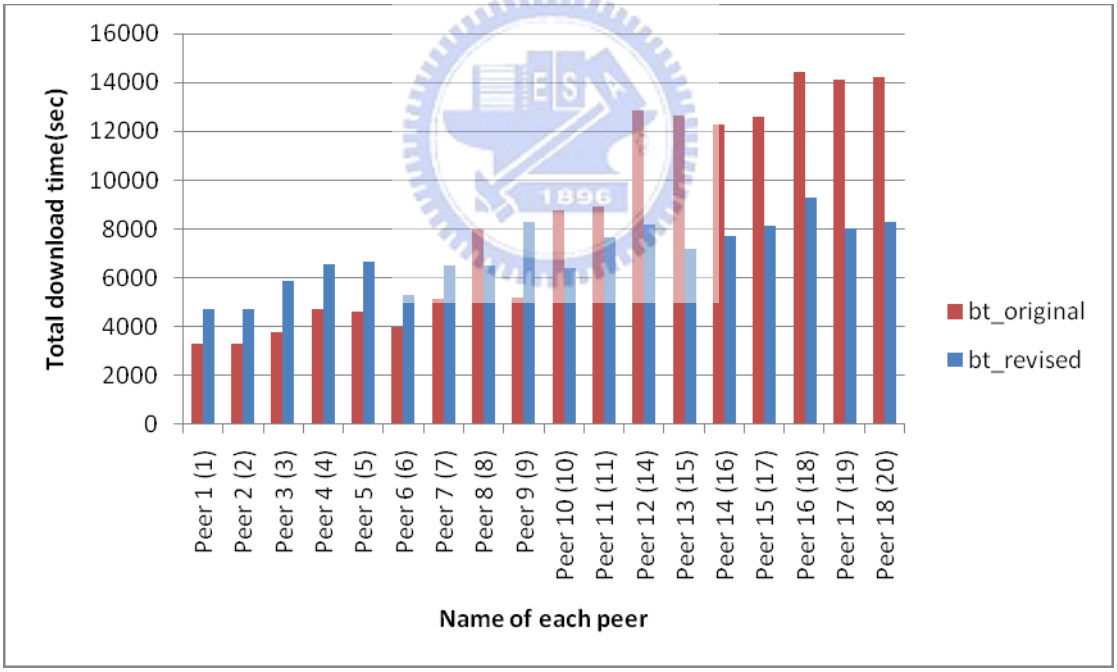


圖表四：Data Diffusion (Standard Deviation of Owner Count)

- **同儕下載檔案個別花費時間：**如圖表五所示，每組長條圖都是由同一個下載者產生，左邊的長條圖為原本的 BT，而右邊的長條圖則是我們修改的 BT*，長條圖長度越長代表下載檔案的時間越久。我們可以從中發現雖然上傳頻寬較大的 BT*的下載者花了比原本的 BT 還久的時間下載檔案，但是上傳頻寬

較小的 BT*的下載者卻比原本的 BT 減少了許多時間下載檔案。表格 2 顯示出每組不同上傳頻寬的 BT*下載者與原本的 BT 下載者個別平均所花的下載時間的差距，從這個表格中可以很清楚地看見上傳頻寬高的 BT*下載者雖然比原本的 BT 下載者慢了 1700 秒鐘左右，卻使得上傳頻寬低的 BT*使用者比起原本的 BT 下載者快了 4300 秒鐘，此點的影響力實在是不容忽視。

- **結論：**這是因為我們所修改的演算法總是有幫助比較弱勢的使用者，也就是完成度高的同儕會幫助那些完成度低的同儕，因此高上傳頻寬的同儕不會總是與其他高上傳頻寬的使用者分享資料，所以我們可以了解為什麼在 BT*內上傳頻寬高的同儕會比原本使用 BT 的下載時間來的久；反觀上傳頻寬較低的使用者卻因為高上傳頻寬使用者的幫助而大幅減少下載的時間。整體的平均下載時間增快了 17.62%。



圖表五：Session Duration(Peer Completion Time)(左邊為 bt_revised、右邊為 bt_original)

表格 2：Session Duration Gain /Loss of each group

Upload Rate	80KB/s	60KB/s	40KB/s	20KB/s	10KB/s	5KB/s
Session Duration Gain/Loss(sec)	1632.667	1722	976	-2578.67	-4816.67	-5742.33

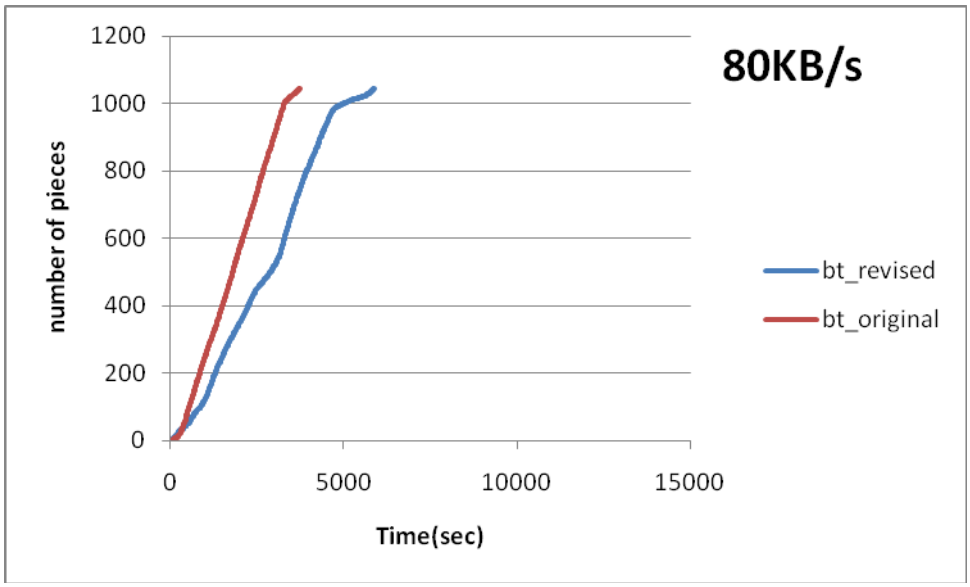
- **各個組別的同儕下載檔案的詳細情形：**圖表六到圖表十一是我們將每組的資

料個別平均所得到的資料累積情形圖，我們可以從這幾張圖看出兩個現象。第一，我們似乎可以看到 BT*在每個組別前 500 秒的曲線斜率好像差不多，這代表 BT*在大家都沒有東西的情形下可以很快的散播資料，我們可以從圖表十二看出此現象，因為大家一開始都沒東西，當一開始與擁有完整資料的做種者連線的同儕拿到一塊檔案片段時，他們就會立即散播給那些沒有任何檔案片段的人，當大家拿的東西越來越多，時間越來越長，演算法的另一重點開始產生效用，也就是以牙還牙，因此大家拿取資料的速度開始有了差異；圖表十二與圖表十三比較之後更能驗證我們所說的是正確，因為原本的 BT 只有考量以牙還牙的政策，若是在一開始大家都沒東西的時候，BT 下載者只能等待其他擁有資料的 BT 下載者利用樂觀選擇上傳同儕這個機制被選擇為可上傳資料的同儕，因而得到一些可與別人交換的資料。但是我們的 BT*會照顧弱勢族群，擁有資料的同儕不需要等到樂觀選擇方法的時間[8]就可以開放讓那些完成度低的同儕拿取資料，所以一開始大家的上傳速率都差不多。第二個現象就是我們可以看到 BT*的資料累積曲線總是有明顯地彎彎曲曲，我們推論是因為下載速率波動過大導致曲線跟原本 BT 的資料累積曲線比起來就不是那麼直順，因此我們從每一組選出一個同儕畫出下載速率的時間對應圖[9]，這些圖分別是從圖表十四至圖表十九，從這些圖中我們的確可以看到 BT*的下載速率的確震盪的很劇烈，而波動這麼大的原因推論是因為有時候某些同儕拿到過於稀有的檔案片段，因此大家都選擇上傳資料給這些同儕，使得這些同儕可以從大家那邊拿到許多資料，等到此檔案片段變成不稀有的情況，則同儕的下載速率馬上跌到谷底。另外，我們也可以看到在高上傳頻寬的組別 BT*的下載速率大部分都是低於原本的 BT，所以很容易就看出來原本的 BT 速度在高上傳頻寬的組別比我們快；在低上傳頻寬的組

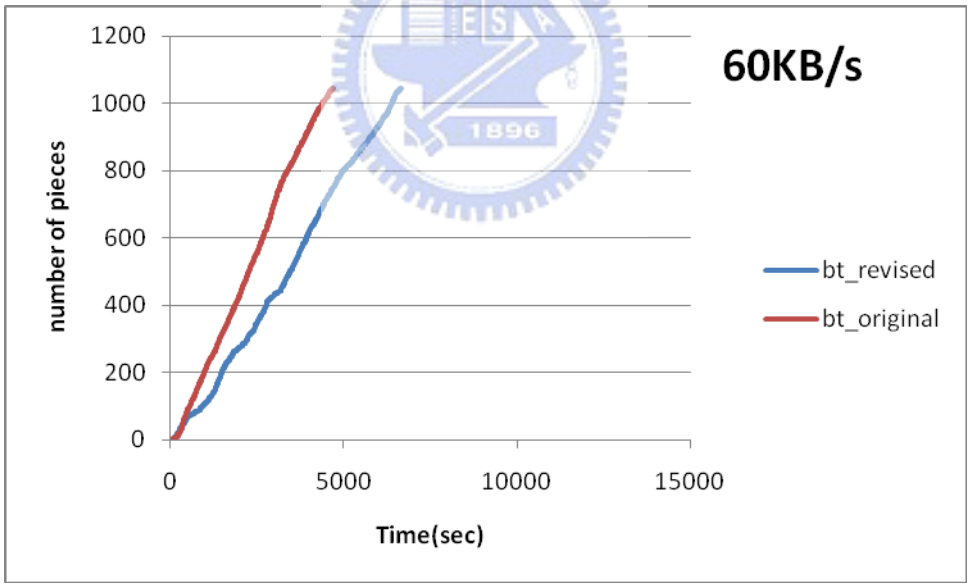
[8] 在 BT 協議內，樂觀選擇方法是每 30 秒決定一次要上傳資料的同儕；在 BT*內，我們是每 10 秒就會決定一次要上傳資料的同儕。

[9]由於我們使用移動平均法(moving average)來畫此圖，所以可以看到比較平順的曲線。

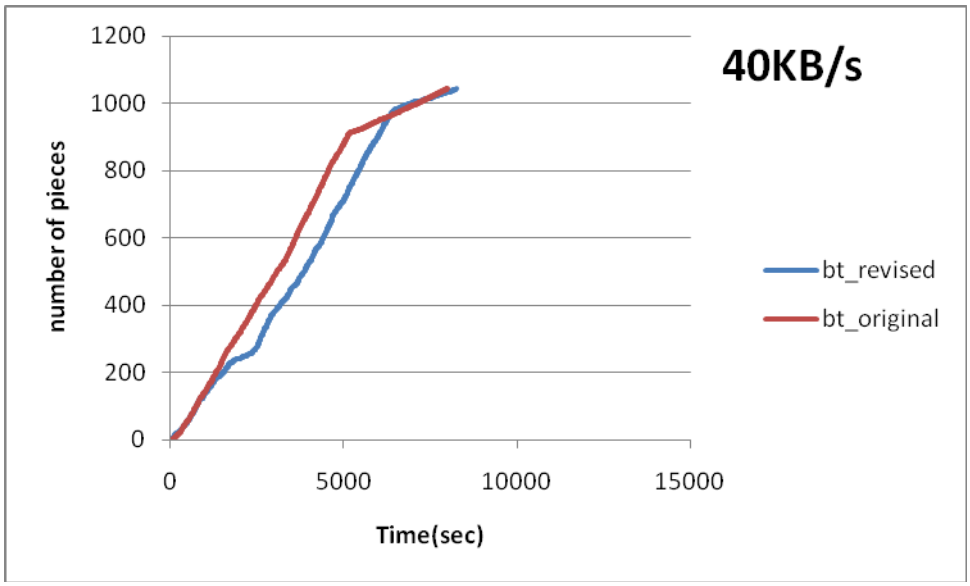
別，BT*的下載速率大部分都高於原本的 BT，因此也可以知道 BT*在低上傳頻寬的組別跑得比原本的 BT 快。



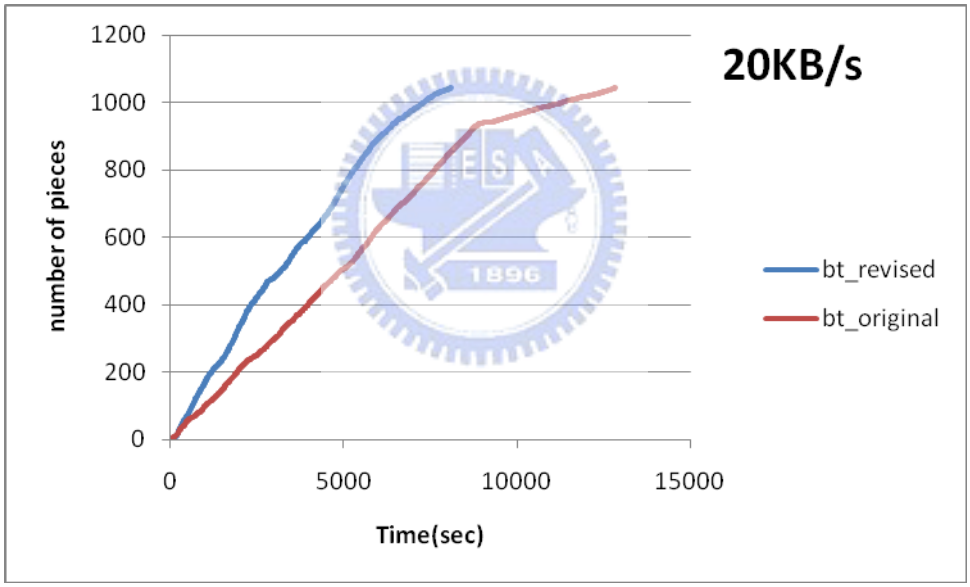
圖表六：Data Accumulation of Peers with Upload Rates 80KBps



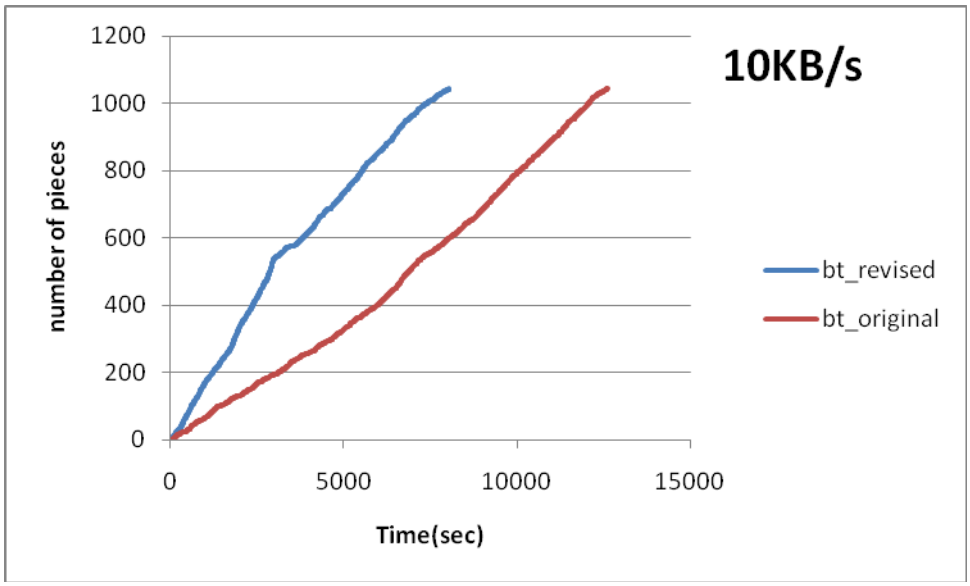
圖表七：Data Accumulation of Peers with Upload Rates 60KBps



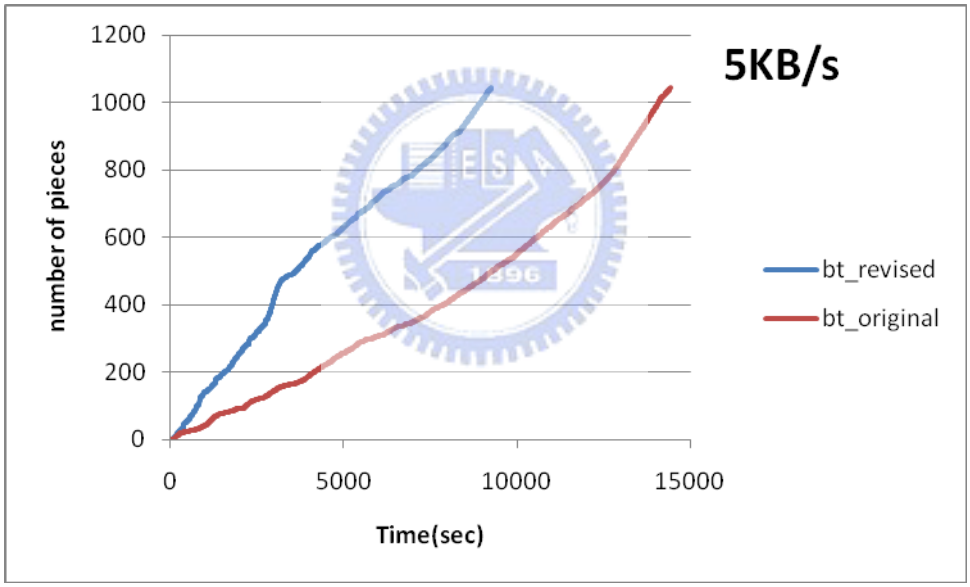
圖表八：Data Accumulation of Peers with Upload Rates 40KBps



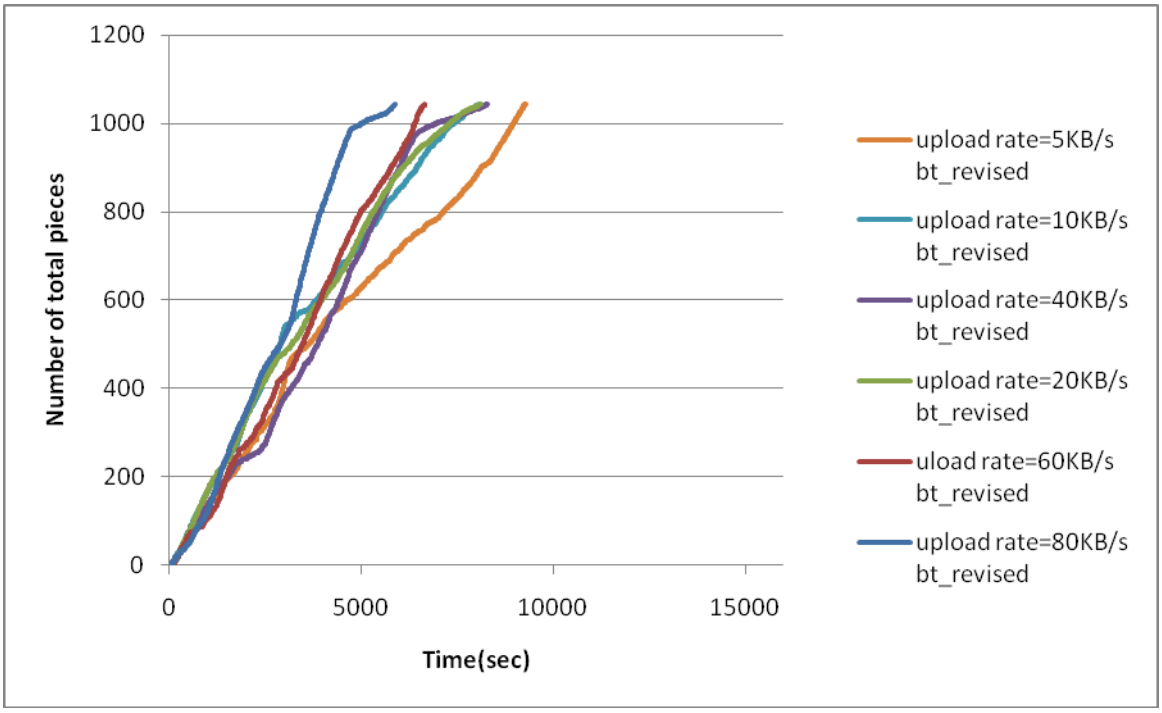
圖表九：Data Accumulation of Peers with Upload Rates 20KBps



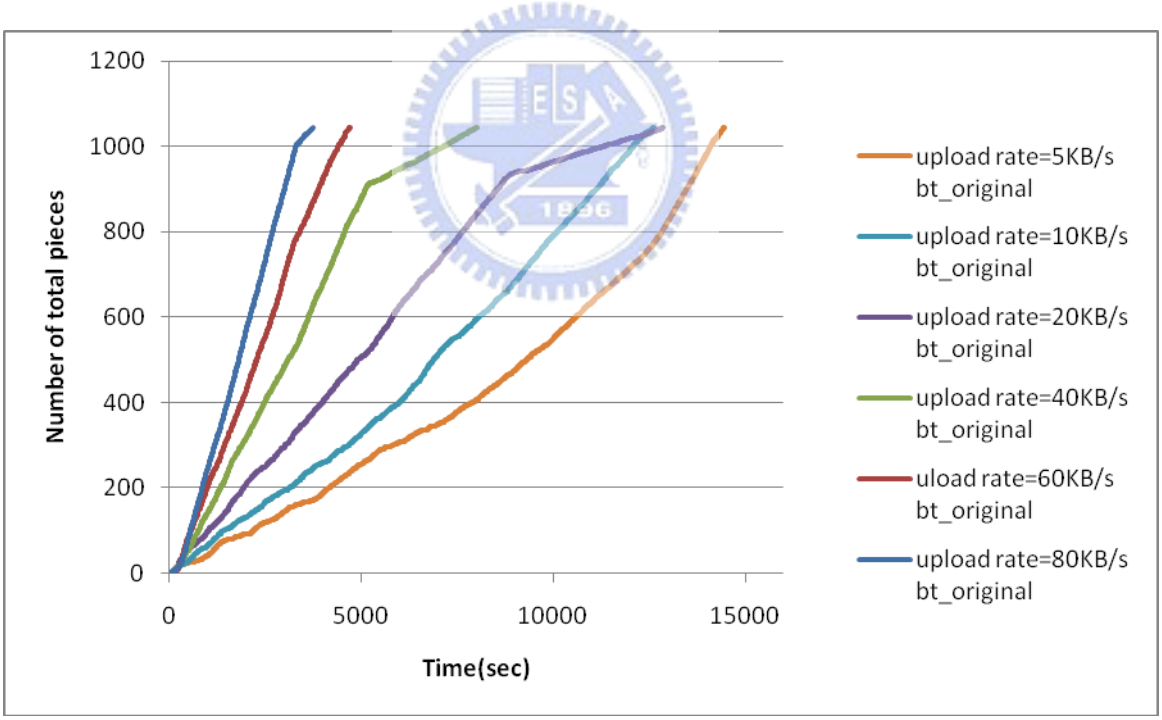
圖表十：Data Accumulation of Peers with Upload Rates 10KBps



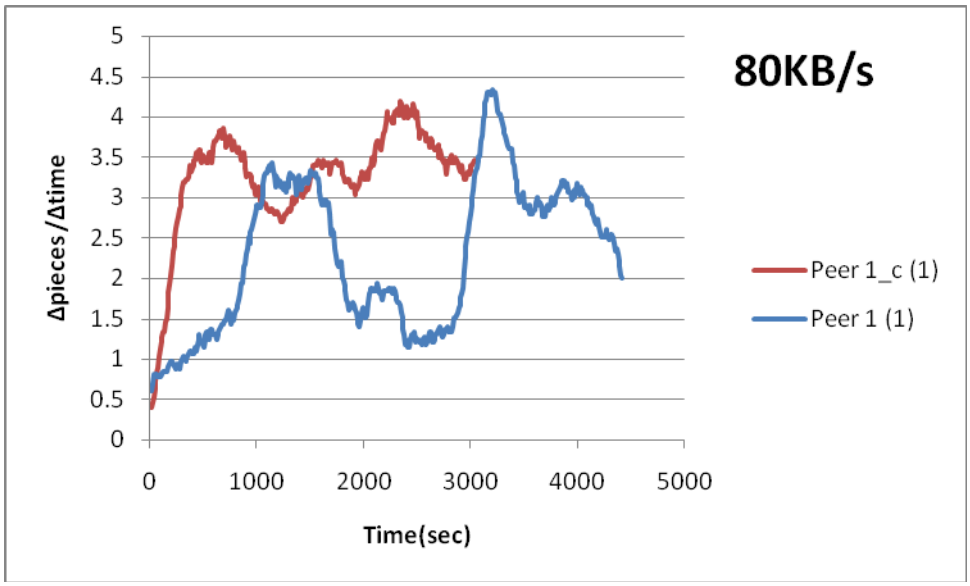
圖表十一：Data Accumulation of Peers with Upload Rates 5KBps



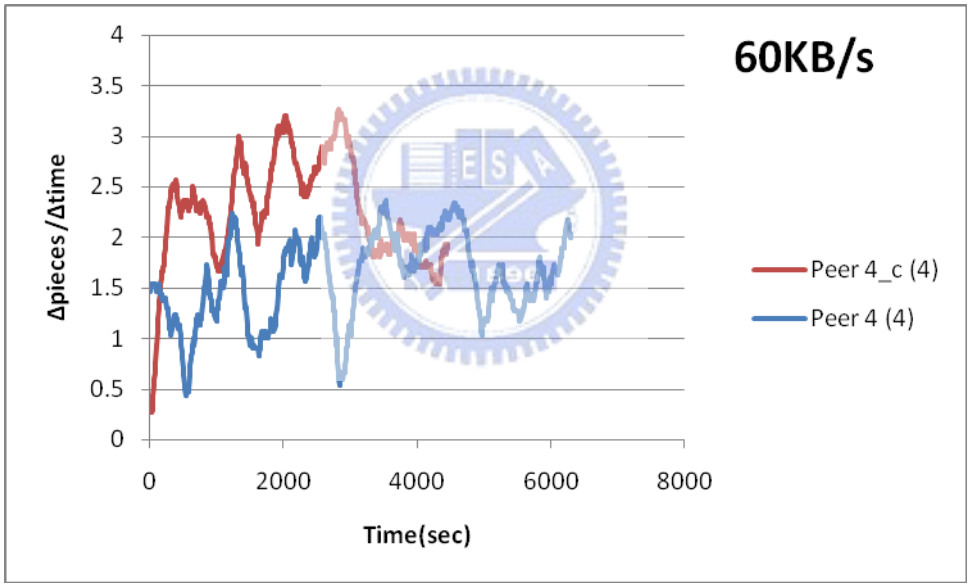
圖表十二：Comparison of Data Accumulation of Peers with Different Upload Rates Using BT*



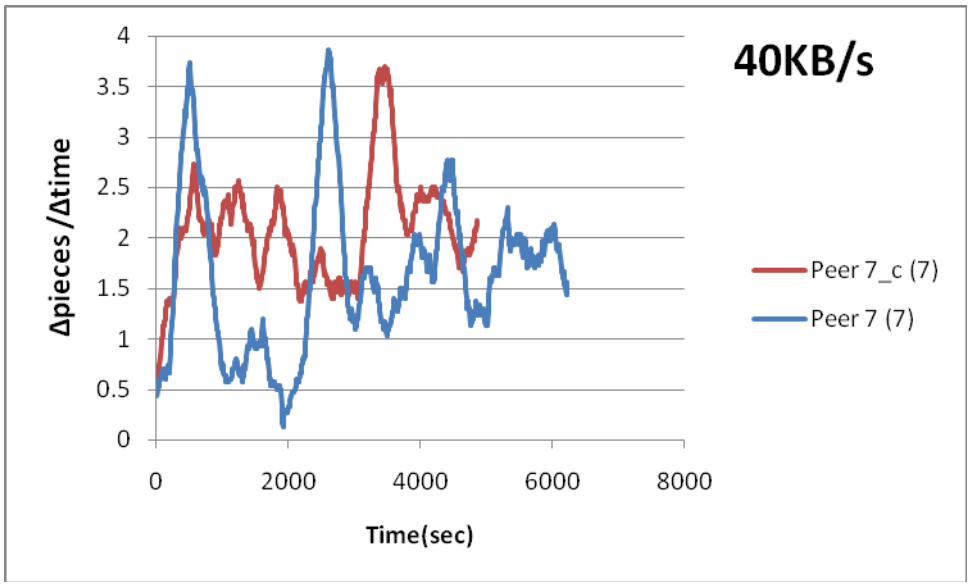
圖表十三：Comparison of Data Accumulation of Peers Using BT



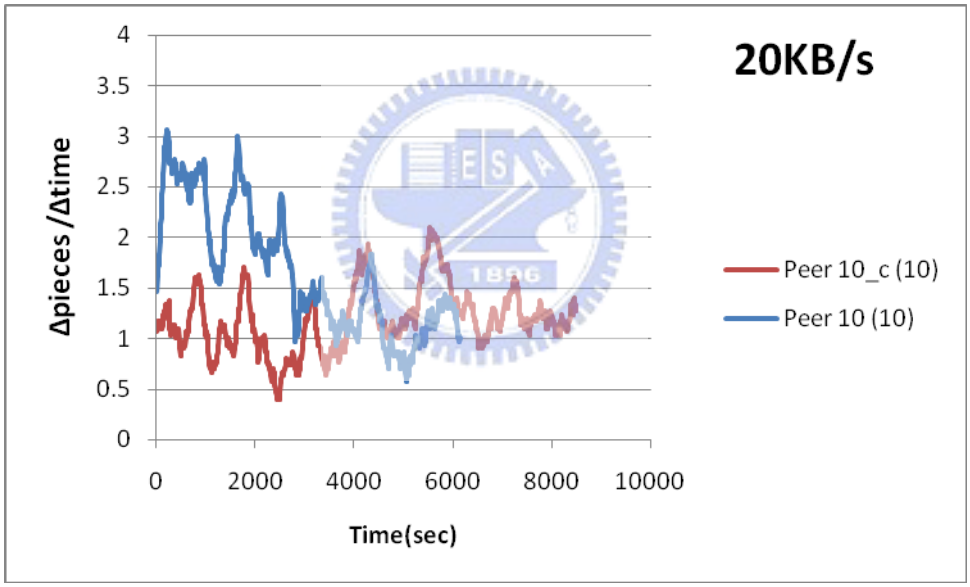
圖表十四：Data Accumulation of Peer1 with Upload Rates 80KBps



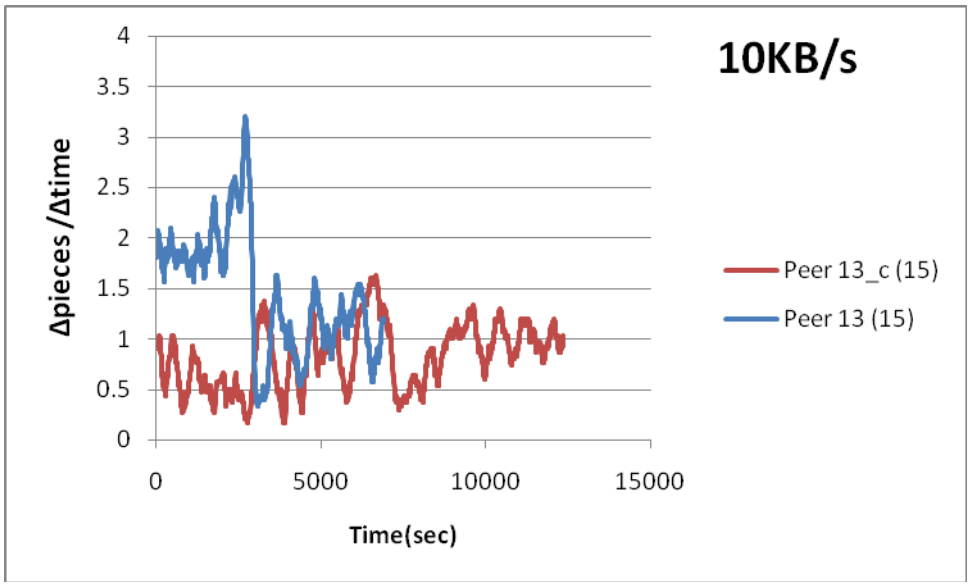
圖表十五：Data Accumulation of Pee4 with Upload Rate 60KBps



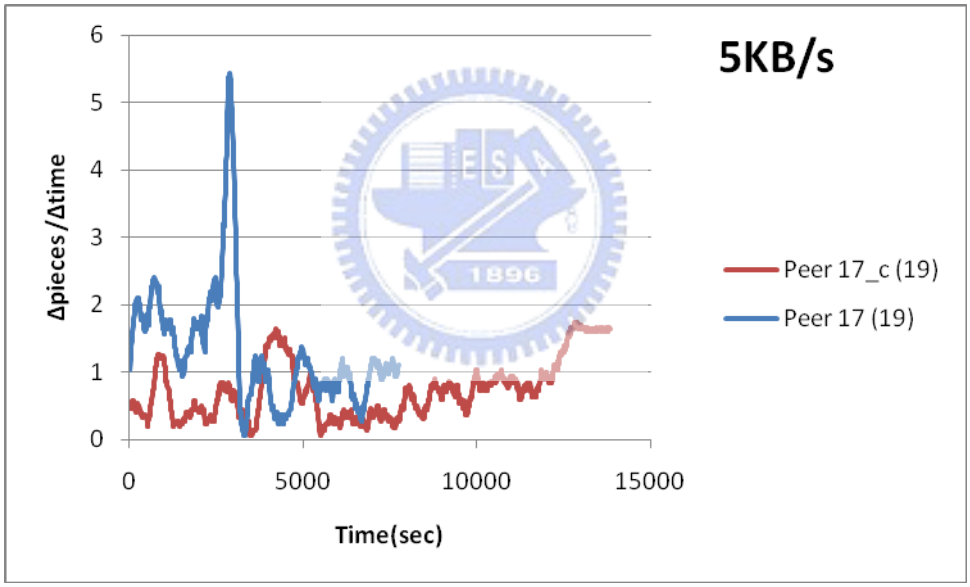
圖表十六：Data Accumulation of Peer7 with Upload Rate 40KBps



圖表十七：Data Accumulation of Peer10 with Upload Rate 20KBps



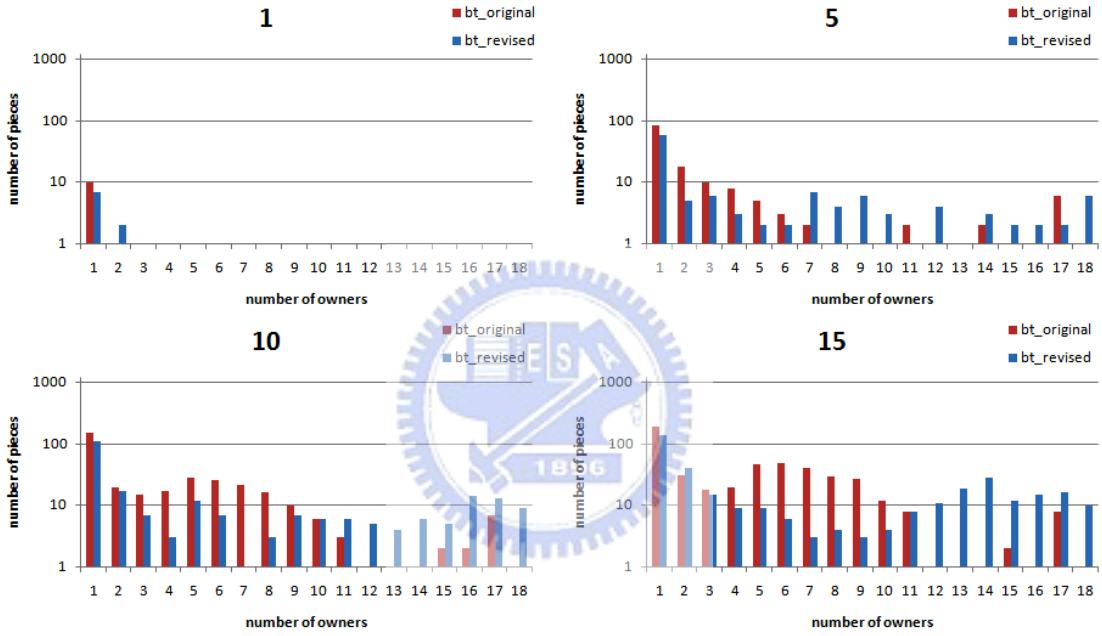
圖表十八：Data Accumulation of Peer13 with Upload Rate 10KBps



圖表十九：Data Accumulation of Peer17 with Upload Rate 5KBps

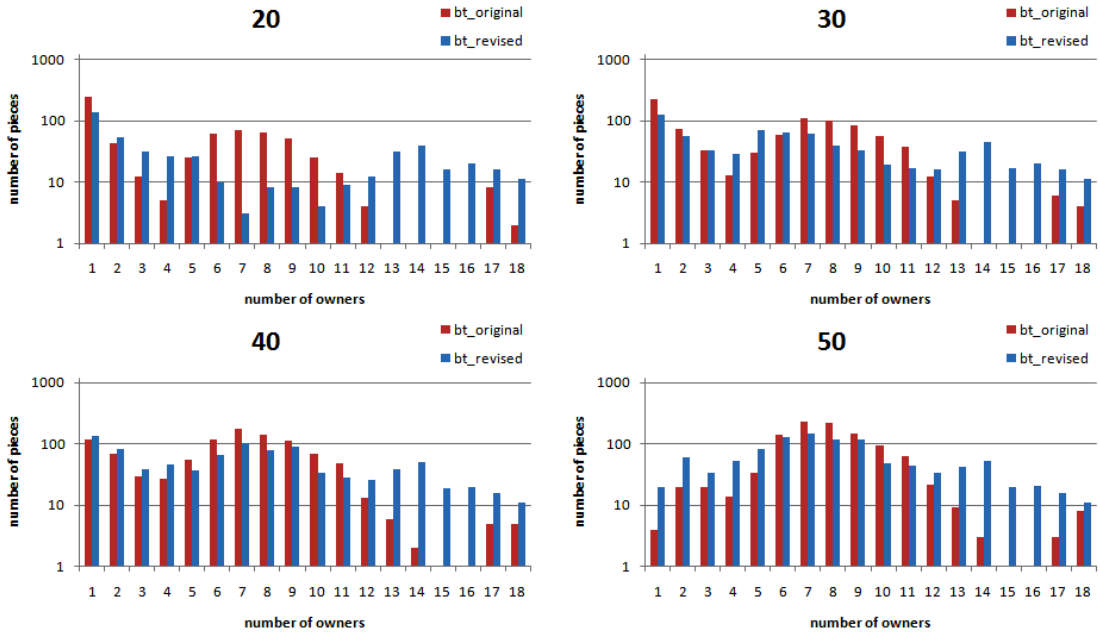
- 整體的資料散播情形：圖表二十到圖表二十四所代表的意義為在每個時間點被 1 個同儕擁有的檔案片段有幾塊、被 2 個同儕擁有的檔案片段有幾塊、被 3 個同儕擁有的檔案片段有幾塊、…、被 18 個同儕擁有的檔案片段有幾塊，長條圖上方的數字代表那個圖是在第幾分鐘統計，我們可以看到在第一分鐘結束的時候，其實 BT*還有一些值是 1 的沒有被顯

示出來[10]，而原本的 BT 卻只有 10 個不同的檔案片段被一個人擁有，所以一開始 BT*的散播速度比原本的 BT 快。按著時間看下去，我們可以發現雖然一開始的散播情況並沒有很明顯的具有優勢，但是 BT*的資料分佈總是可以一直被很快的散播出去，因為我們可以看到高擁有人數的檔案片段數目總是比原本的 BT 多，所以我們還是可以說 BT*的資料散播速度的的確確是比原本的 BT 快。到了第 80 分鐘，BT*的散播速度開始明顯地比原本的 BT 快，自此之後便開始拉大差距直到 BT*結束為止。

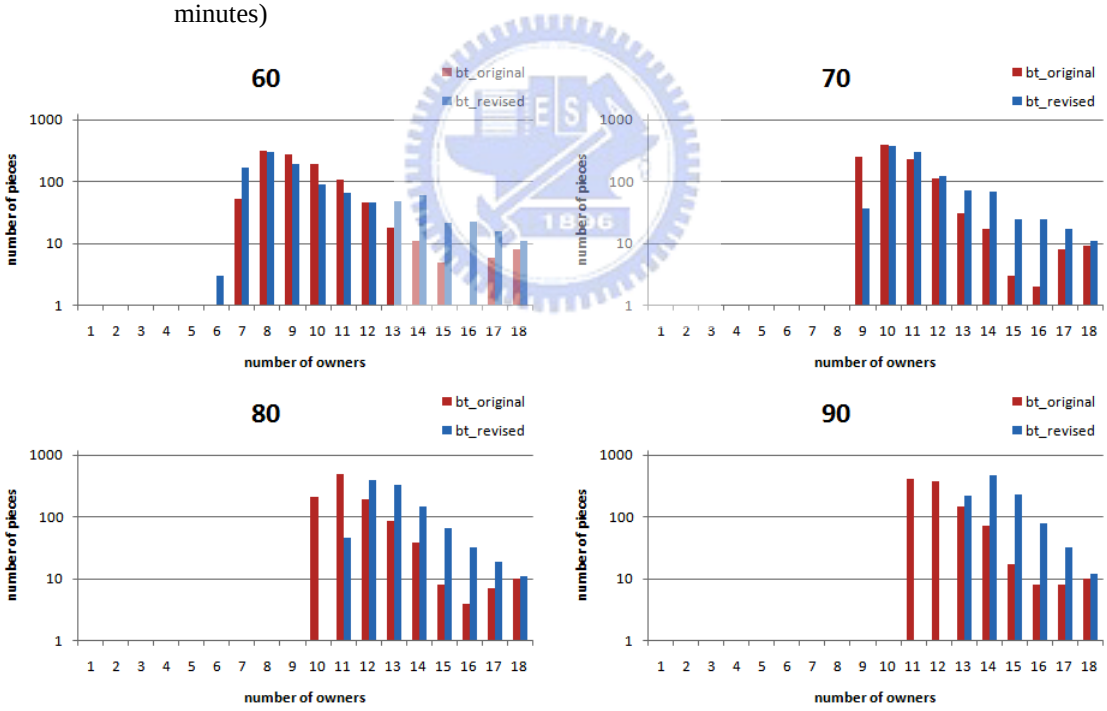


圖表二十：Data Diffusion rates: Pieces Count vs. Owner Count (Start Period, 1 minute – 15 minutes)

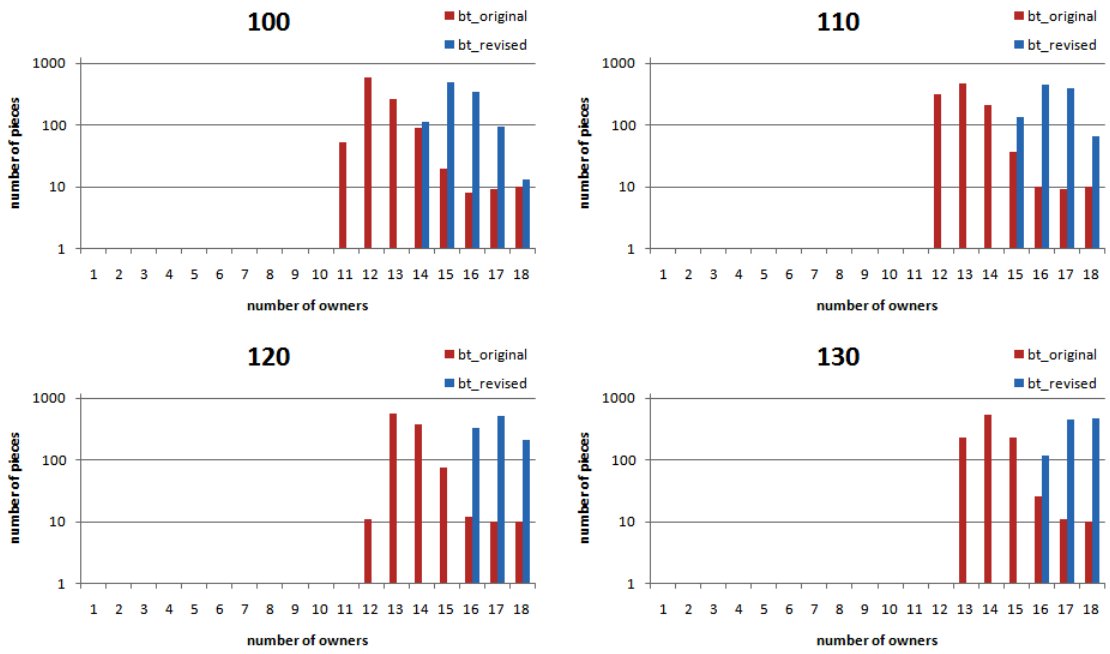
[10] 因為我們將圖的 y 軸刻度改成使用對數刻度基底為 10，因此當值為 1 時剛好是 y 軸最小的刻度，所以無法在此圖中被顯示。



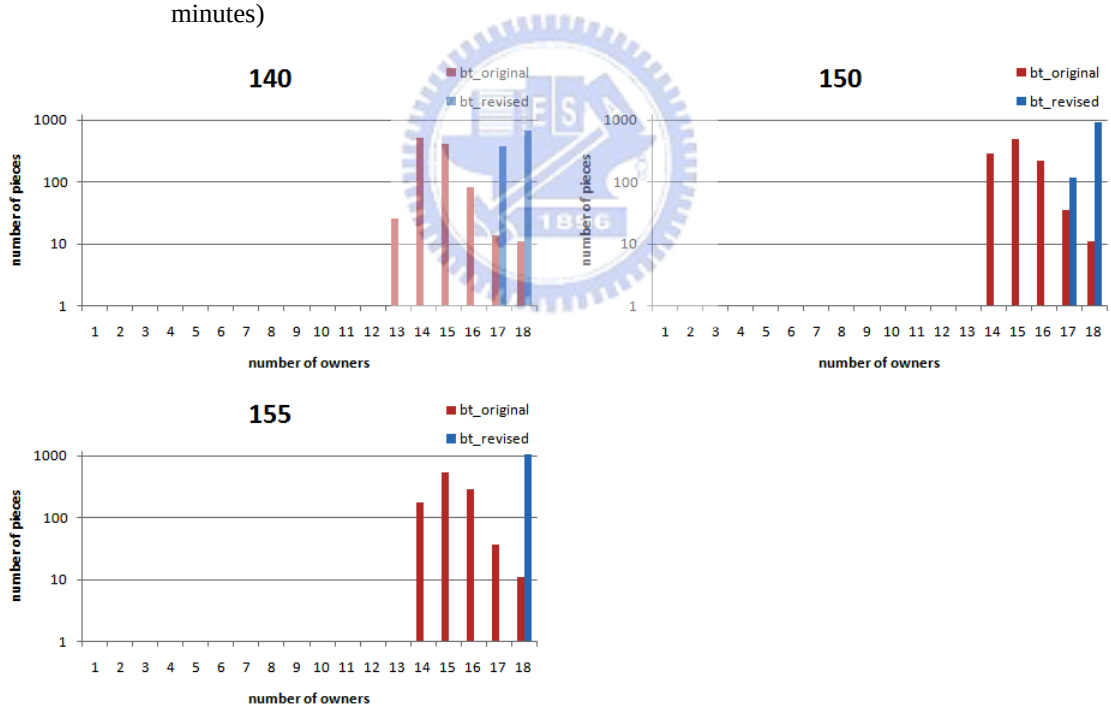
圖表二十一：Data Diffusion rates: Pieces Count vs. Owner Count (Early Period, 20 minutes – 50 minutes)



圖表二十二：Data Diffusion rates: Pieces Count vs. Owner Count (Middle Period, 60 minutes – 90 minutes)



圖表二十三：Data Diffusion rates: Pieces Count vs. Owner Count (Late Period, 100 minutes – 130 minutes)



圖表二十四：Data Diffusion rates: Pieces Count vs. Owner Count (End Period, 140 minutes – 155 minutes)

Chapter 5 Conclusion

在本章節中將會對我們的研究下一個結論，並且探討未來讓此研究變得更好和更加可靠的一些必要工作。

5.1 Accomplishment

在此篇論文中，我們主要著眼於欲改善的兩個重點：第一，對於過於稀有的資訊片段在原本的 BitTorrent Protocol 內並沒有被好好的處理，因為 BitTorrent Protocol 的原則總是為了自身的下載速度著想，上傳頻寬越高的同儕理所當然可以越快的將檔案下載完成，但是當這些上傳頻寬較高的同儕一下載完就離開分享的行列，只剩下頻寬較低的同儕們互相分享，此時就有可能出現某些檔案片段短缺的情形，甚至出現少數檔案片段是無人擁有的，如此一來就沒有人能夠完成檔案的下載。且根據[8]指出速度快的同儕總是在自己的小圈子內分享檔案，因此有可能造成某些檔案片段無法傳到速度慢的同儕手中，也就是我們所謂的過於集中的檔案片段的現象，若是那些速度快的同儕一下載完成就不再分享，還是有可能出現剛剛提到的狀況。第二，因為現行的 BitTorrent Protocol 並無法規定每個下載者必須上傳多少資料或是上傳多少時間之後才能離開，如果每個人都很自私的一下載完就離開，某些速度慢一點的人或是比較晚才開始下載的人可能就沒辦法下載完全的檔案。

我們在此篇論文的貢獻如下：

- 我們提出新的上傳同儕選擇演算法，此演算法利用簡單的加權函數以及隨著下載階段不同而跟著變化的比重權值在不同的階段使用新的上傳同儕選擇機制。
- 我們對於檔案片段的分散性也提出一個簡單的機制來避免一些檔案片段的遺失，利用高低水線的概念控制下載情況的緊急度。

- 我們已經在 Azureus 內實現所提出的這些演算法，隨時可以上傳至 Azureus 討論區供人下載。
- 我們利用更改之後的 Azureus 實地跑了許多次實驗，得到的結果如下：
  - 檔案片段的散播更穩定暨更快速的散播。
  - 一開始的檔案散播速度快於原本使用的上傳同儕選擇演算法。
  - 整體下載速度增快 17.6%。

5.2 Future Work

此研究仍然有一些未做的工作，這些工作敘述如下：

- 利用更多的電腦和不同的網路型態來進行實驗，或是使用大型的網路模擬軟體來模擬更多不同的同儕類型且擴大模擬環境，或許我們所設計的演算法的效果能夠更清楚地顯現。
- 由於目前的實驗沒有使用超過 50 個節點，因此我們不能測試當發生緊急情況時，必須選擇剔除一些同儕，使得 BT 客戶端可以留下空位找新的節點，因而有機會舒緩緊急情況。因此必須藉由網路模擬軟體模擬大型網路或是使用超過 50 台的電腦才有機會得到相關數據。

Reference

[1] Bram Cohen, “Incentives Build Robustness in BitTorrent,” in Workshop on Economics of Peer-to-Peer Systems, 2003.

[2] http://bittorrent.org/index.html

[3] http://azureus.sourceforge.net/

[4] AR Bharambe, C Herley, VN Padmanabhan, “Analyzing and Improving BitTorrent Performance,” Microsoft Research, Feb. 2005.

[5] A Legout, G Urvoy-Keller, P Michiardi, “Rarest First and Choke Algorithms are Enough,” in Proceedings of the 6th ACM SIGCOMM on Internet measurement, 2006.

[6] F Mathieu, J Reynier, M FT R&D, “Missing Piece Issue and Upload Strategies in Flashcrowds and P2P-assisted Filesharing,” in AICT-ICIW, 2006.

[7] Chi-Jen Wu, Cheng-Ying Li and Jan-Ming Ho, “Improving the Download Time of BitTorrent-like Systems,” Institute of Information Science Academia Sinica, Taipei, Taiwan.

[8] N Liogkas, R Nelson, E Kohler, L Zhang, “Exploiting BitTorrent For Fun (But Not Profit),” in Proc. 5th Itl. Workshop on Peer-to-Peer Systems, 2006.

[9] http://wiki.theory.org/Main_Page

[10] X Zhang, J Liu, B Li, TSP Yum, “CoolStreaming/DONet: A Data-Driven Overlay Network for Efficient Live Media Streaming,” in Proceedings of IEEE INFOCOM, 2005.

[11] Y Liu, X Liu, L Xiao, LM Ni, X Zhang, “Location awareness in unstructured peer-to-peer systems,” in IEEE Transactions on Parallel and Distributed Systems, 2005.

[12] AJ Ganesh, AM Kermarrec, L Massoulie, “Peer-to-peer membership management for gossip-based protocols,” in IEEE Transactions on Computers, 2003.

[13] DK Liu, RH Hwang, “P2broadcast: A hierarchical clustering live video streaming system for P2P networks,” in ICCCN, Oct, 2003.

[14] P Maymounkov, D Mazieres, “Kademlia: A peer-to-peer information system based on the XOR metric,” in IPTPS, Mar. 2002.

[15] K Lua, J Crowcroft, M Pias, R Sharma, S Lim, “A survey and comparison of peer-to-peer overlay network schemes,” in Communications Surveys & Tutorials, IEEE, 2005.

[16] A Legout, G Urvoy-Keller, P Michiardi, “Understanding BitTorrent: An Experimental Perspective,” INRIA Sophia Antipolis/INRIA Rhne-Alpes-PLANETE INRIA France, 2005.

[17] SGM Koo, CSG Lee, K Kannan, “A genetic-algorithm-based neighbor-selection strategy for hybrid peer-to-peer networks,” Computer Communications and Networks, 2004.

[18] K Tamilmani, V Pai, A Mohr, “SWIFT: A system with incentives for trading,” in Proceedings of Second Workshop on Economics of Peer-to-Peer, 2005.

[19] M Izal, G Urvoy-Keller, EW Biersack, P Felber, A Hamra, L Garces-Erice, “Dissecting BitTorrent: Five Months in a Torrent's Lifetime,” Passive and Active Measurements, 2004.

[20] D Qiu, R Srikant, “Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks,” in SIGCOMM’04, Aug. 2004.

[21] KP Birman, M Hayden, O Ozkasap, Z Xiao, M Budiu, Y Minsky, “Bimodal Multicast,” in ACM Transactions on Computer Systems, 1999.

[22] R VAN RENESSE, KP BIRMAN, W VOGELS, “Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining,” in ACM Transactions on Computer Systems, 2003.

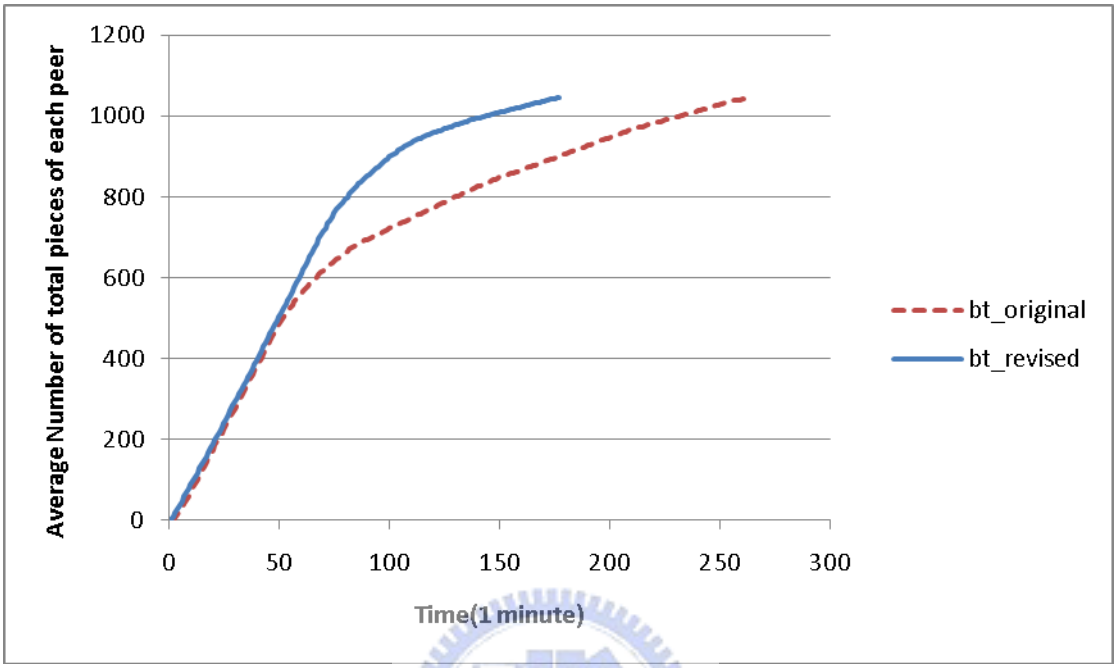
[23] LF Cabrera, MB Jones, M Theimer, “Herald: Achieving a global event notification service,” in HotOS VIII, 2001.

[24] FM Cuenca-Acuna, C Peery, RP Martin, TD Nguyen, “PlanetP: using gossiping to build content addressable peer-to-peer information sharing communities,” in High Performance Distributed Computing, 2003.

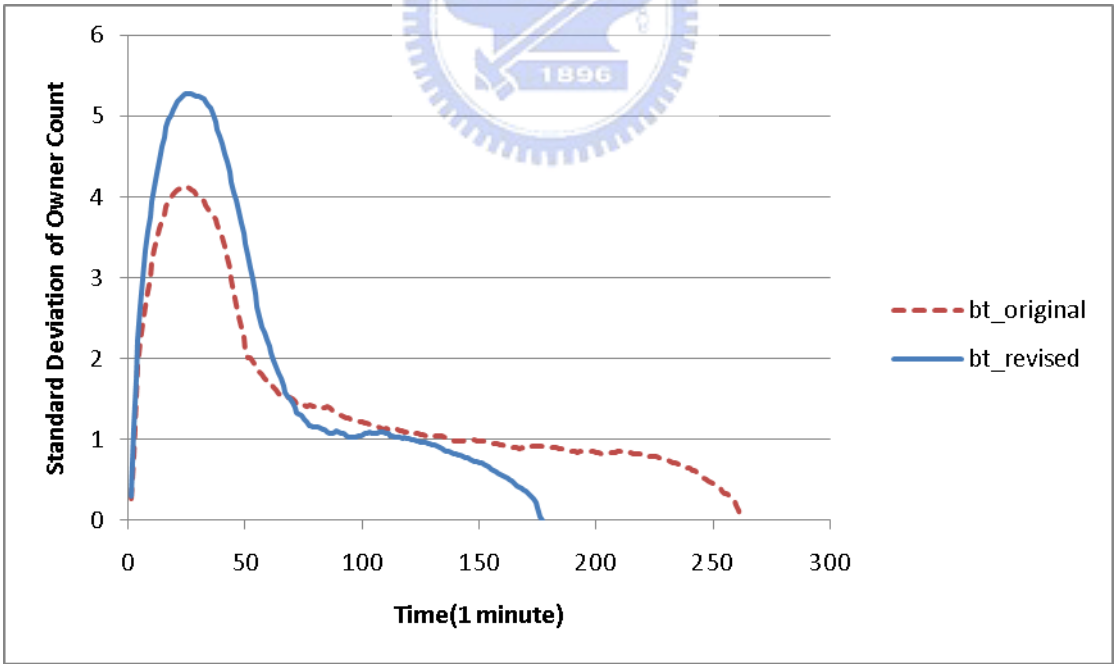
[25] Khambatti, M., Ryu, K., Dasgupta, “Push-Pull Gossiping for Information Sharing in Peer-to-Peer Communities,” in Int’l Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA), (Las Vegas, NV, 2003)

[26] W Vogels, R van Renesse, K Birman, “The Power of Epidemics: Robust Communication for Large-Scale Distributed Systems,” in Proc.of HotNets-I, Computer Communication Review, 33(1), January 2003.

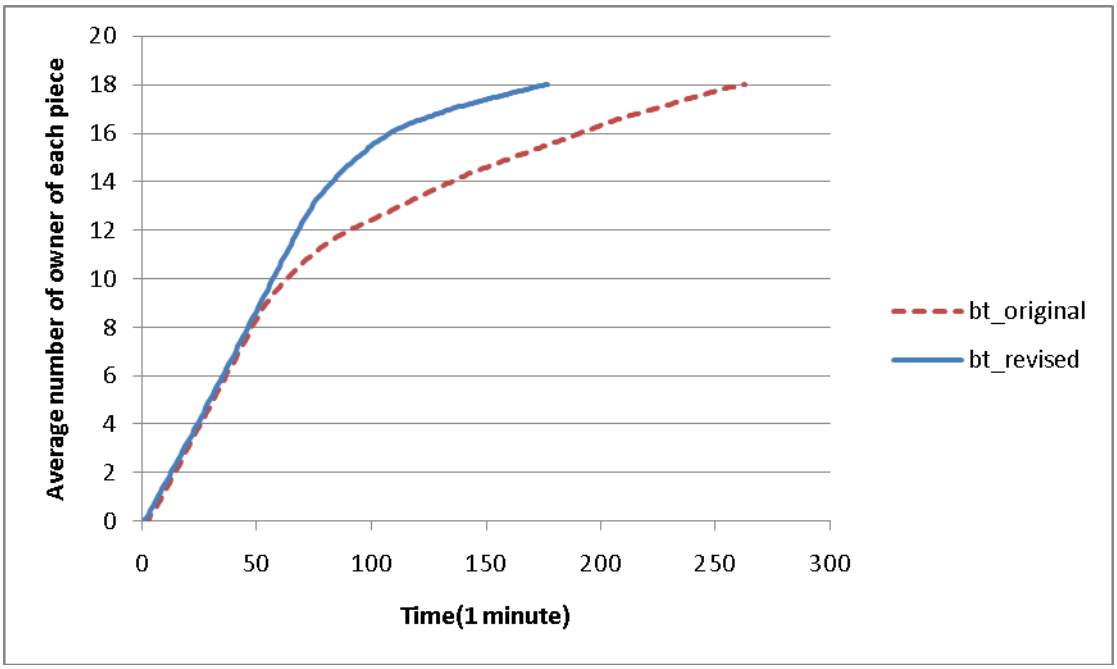
Appendix A (Results of Aug. 14)



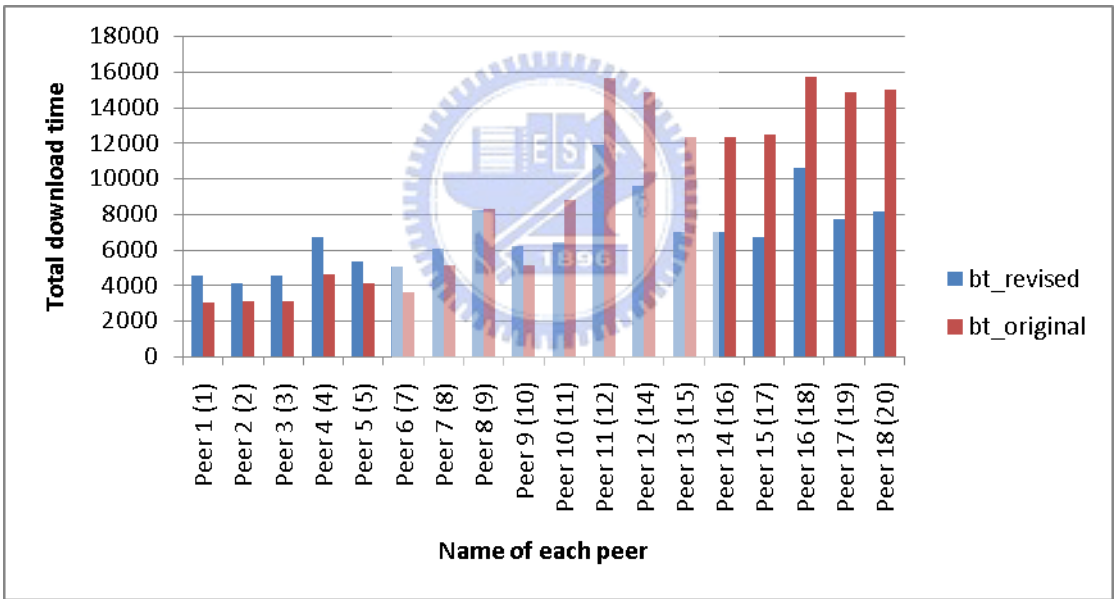
圖表 1：Data Accumulation (Avg. Piece / Peer)



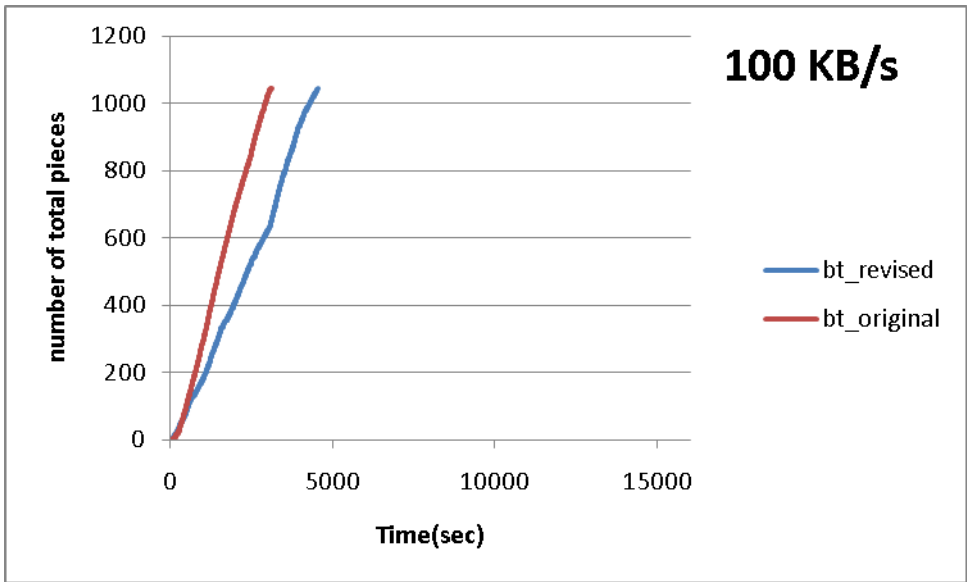
圖表 2：Data Diffusion (Standard Deviation of Owner Count)



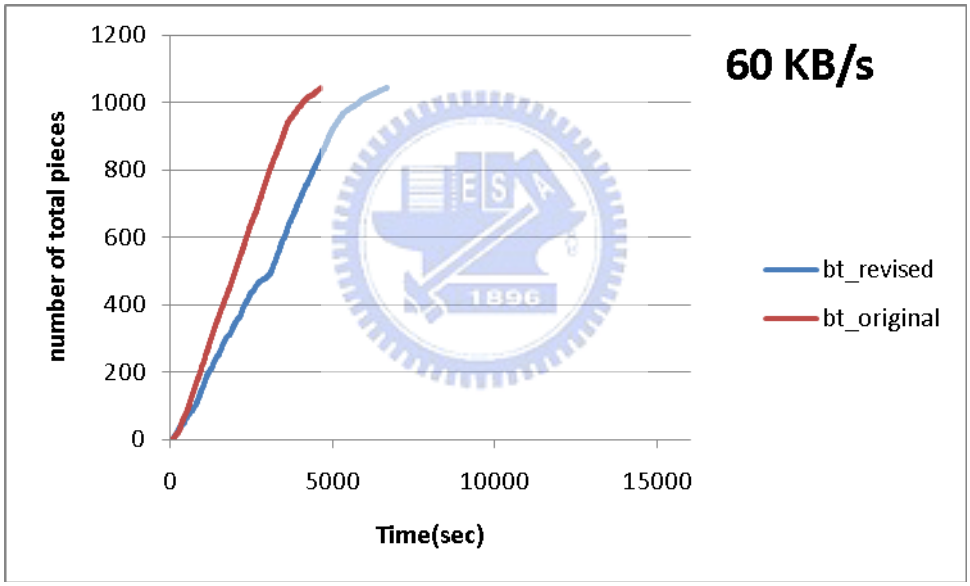
圖表 3：Data Diffusion (Avg. Piece / Peer)



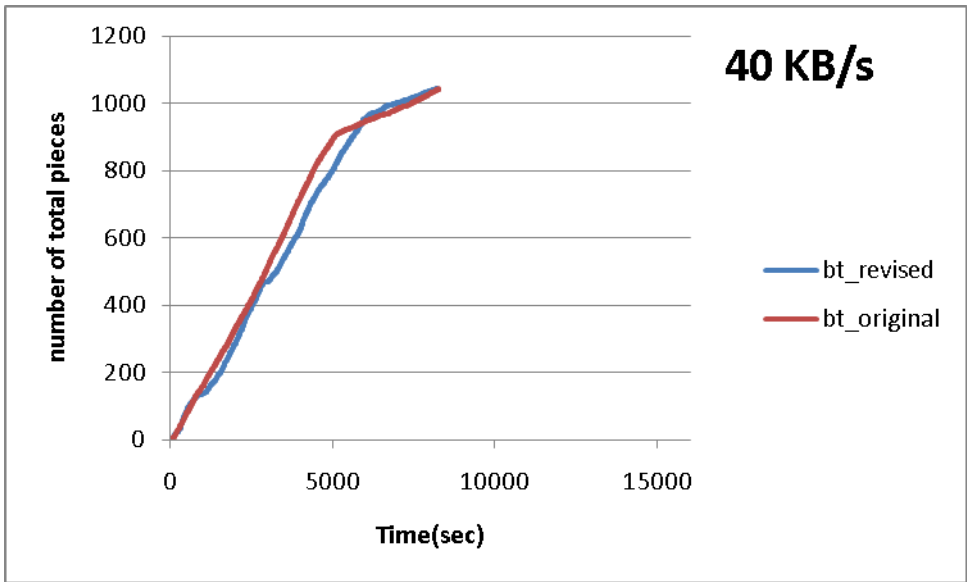
圖表 4：Session Duration (Peer Completion Time)



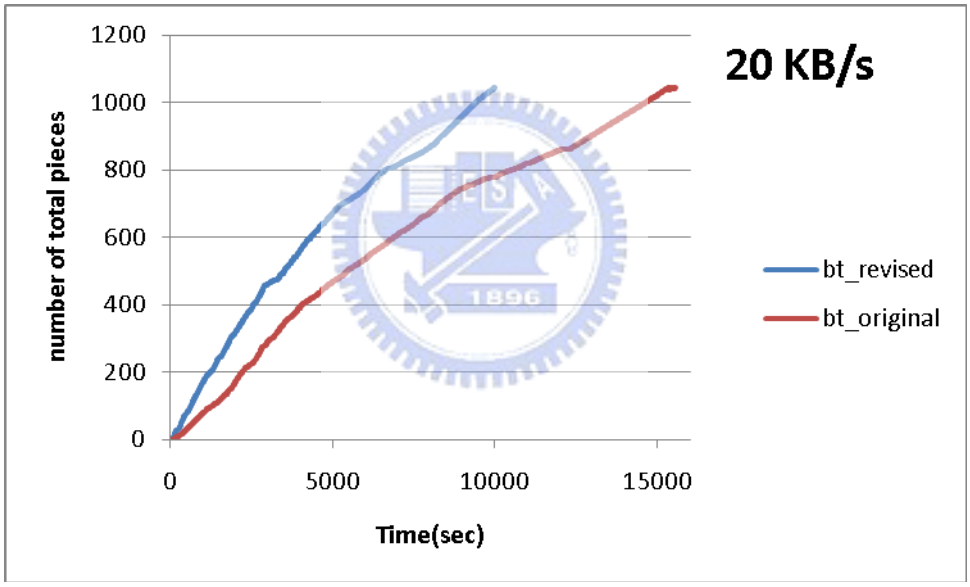
圖表 5：Data Accumulation of Peers with Upload Rates 100KBps



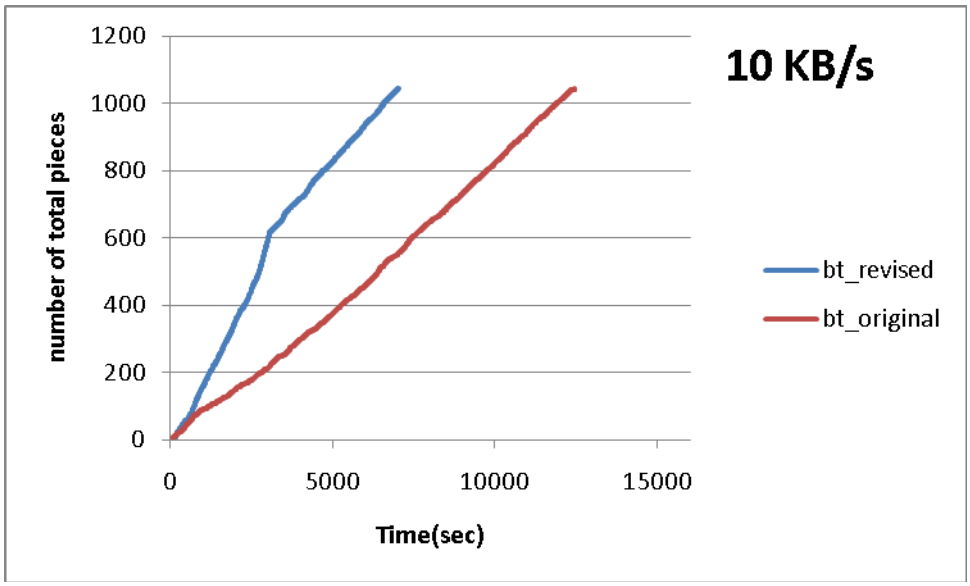
圖表 6：Data Accumulation of Peers with Upload Rates 60KBps



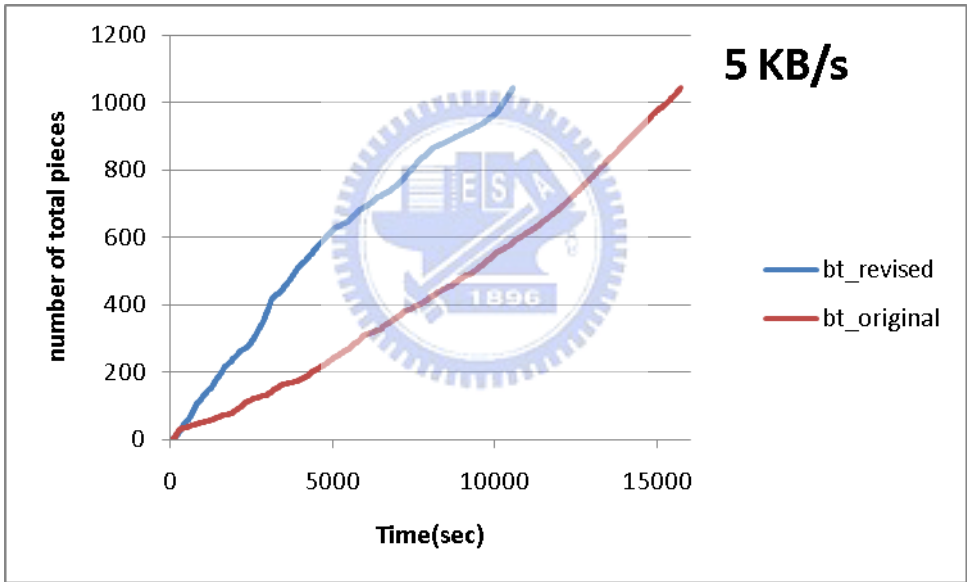
圖表 7：Data Accumulation of Peers with Upload Rates 40KBps



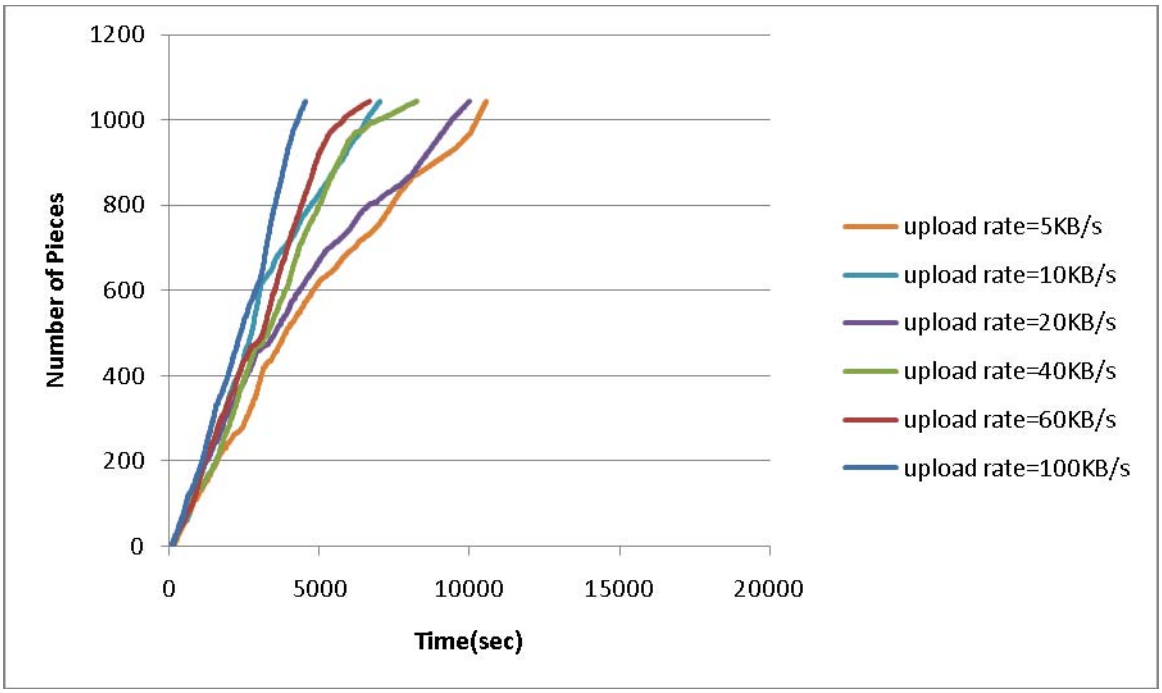
圖表 8：Data Accumulation of Peers with Upload Rates 20KBps



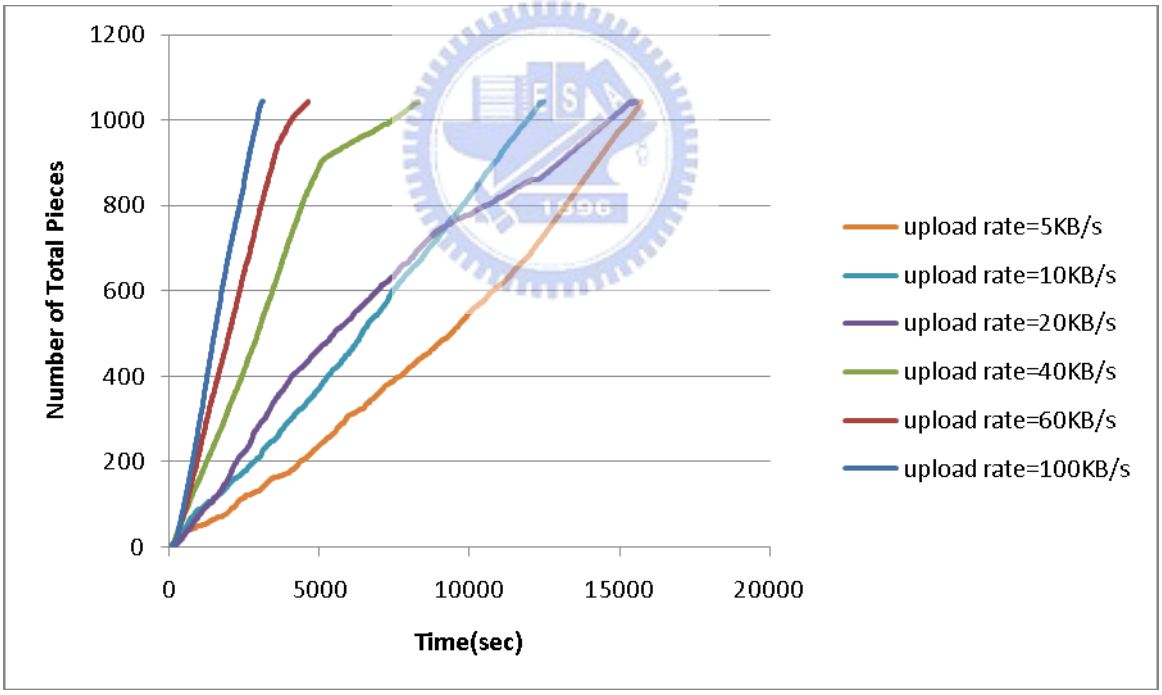
圖表 9：Data Accumulation of Peers with Upload Rates 10KBps



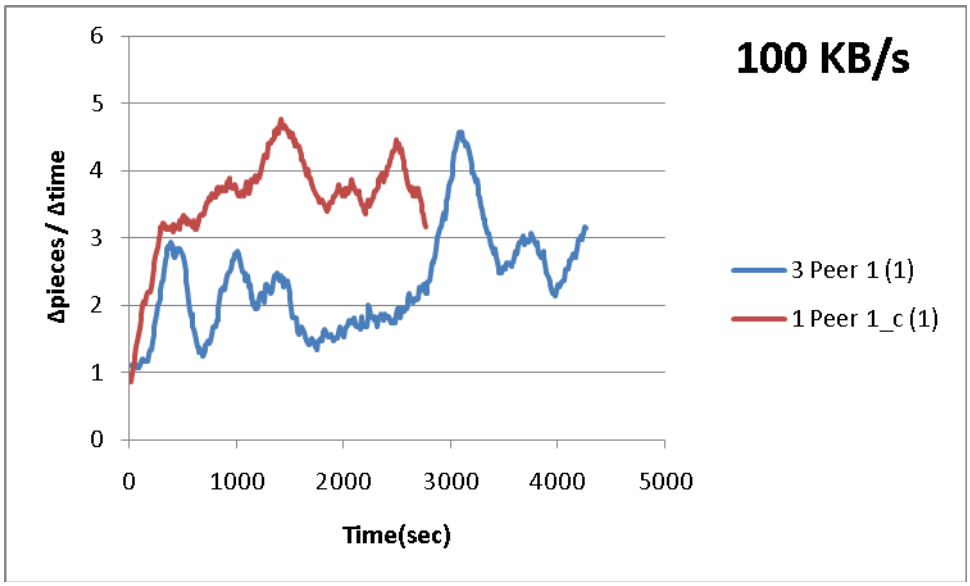
圖表 10：Data Accumulation of Peers with Upload Rates 5KBps



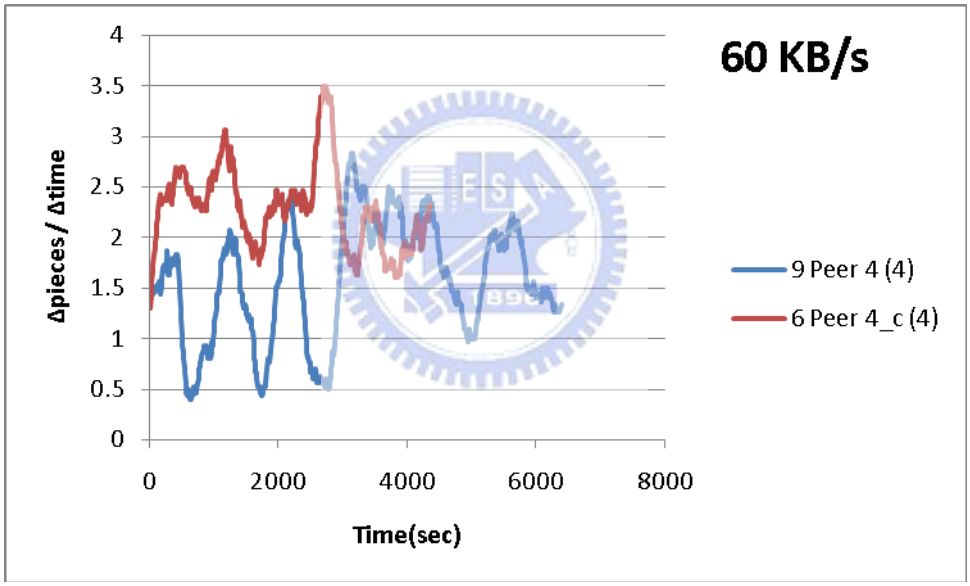
圖表 11：Comparison of Data Accumulation of Peers Using BT*



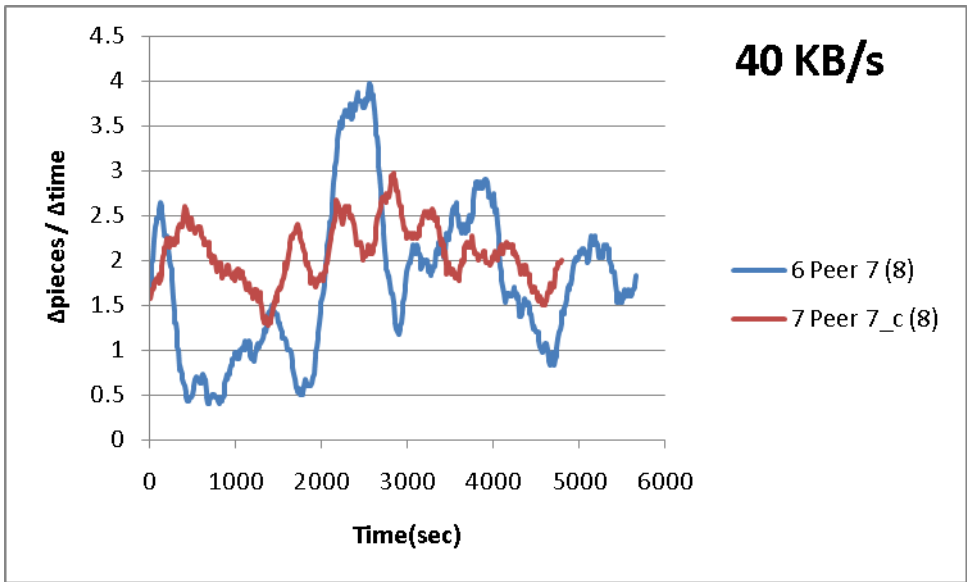
圖表 12：Comparison of Data Accumulation of Peers Using BT



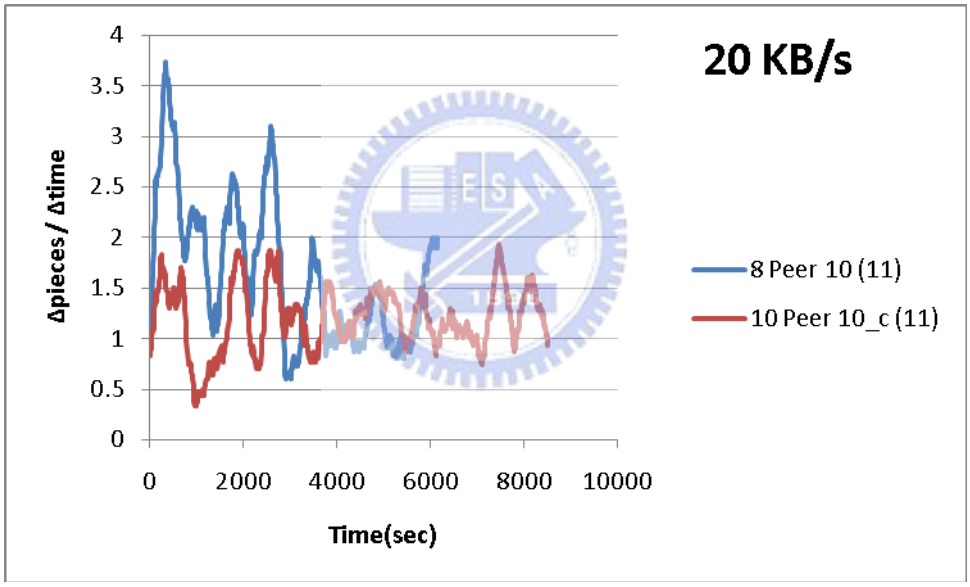
圖表 13：Data Download Rates of Peers with Upload Rates 100KBps



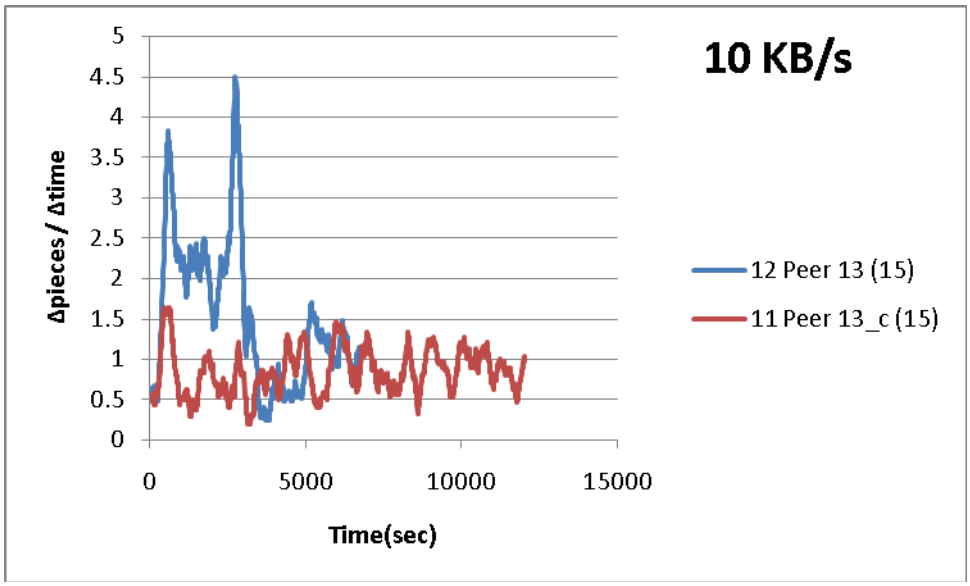
圖表 14：Data Download Rates of Peers with Upload Rates 60KBps



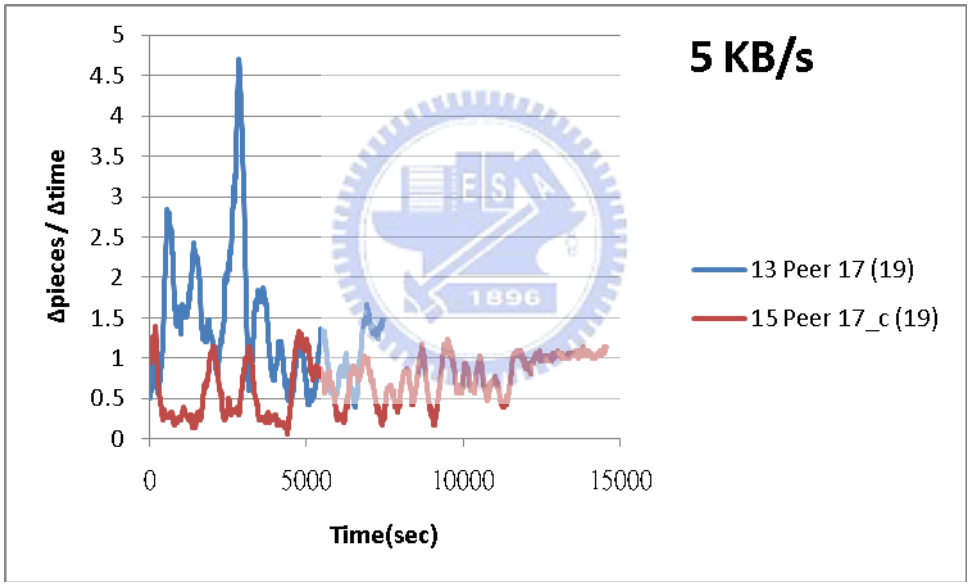
圖表 15：Data Download Rates of Peers with Upload Rates 40KBps



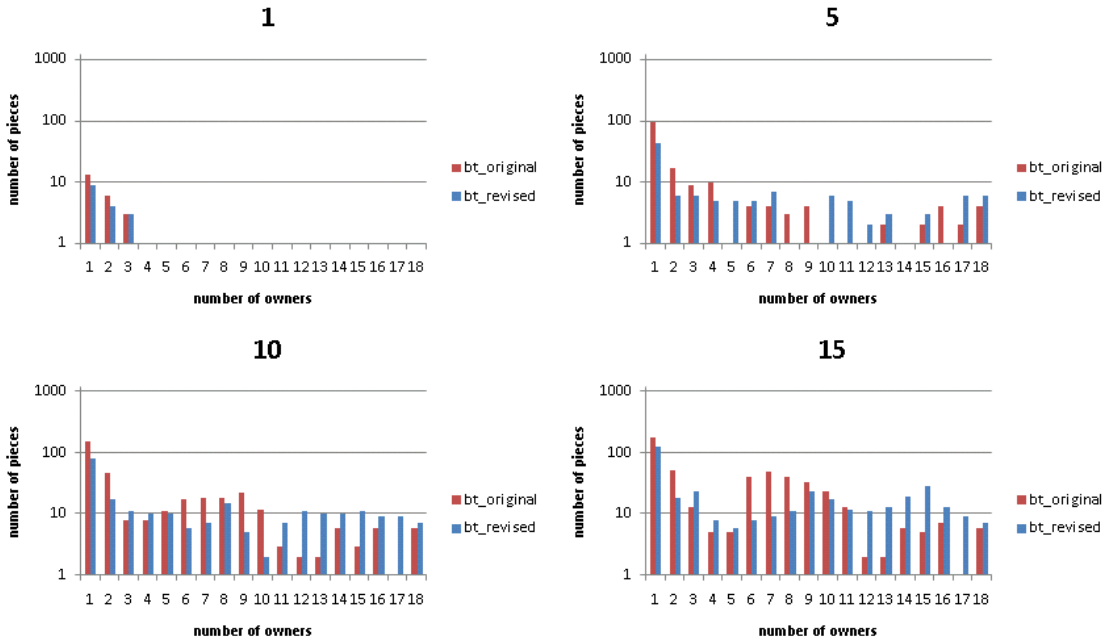
圖表 16：Data Download Rates of Peers with Upload Rates 20KBps



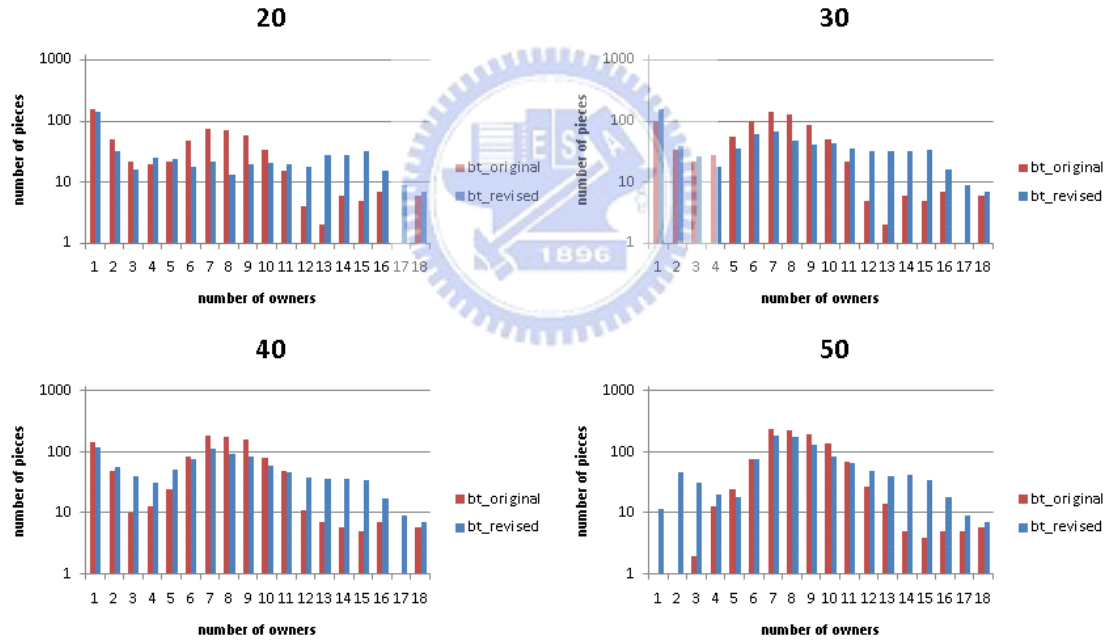
圖表 17：Data Download Rates of Peers with Upload Rates 10KBps



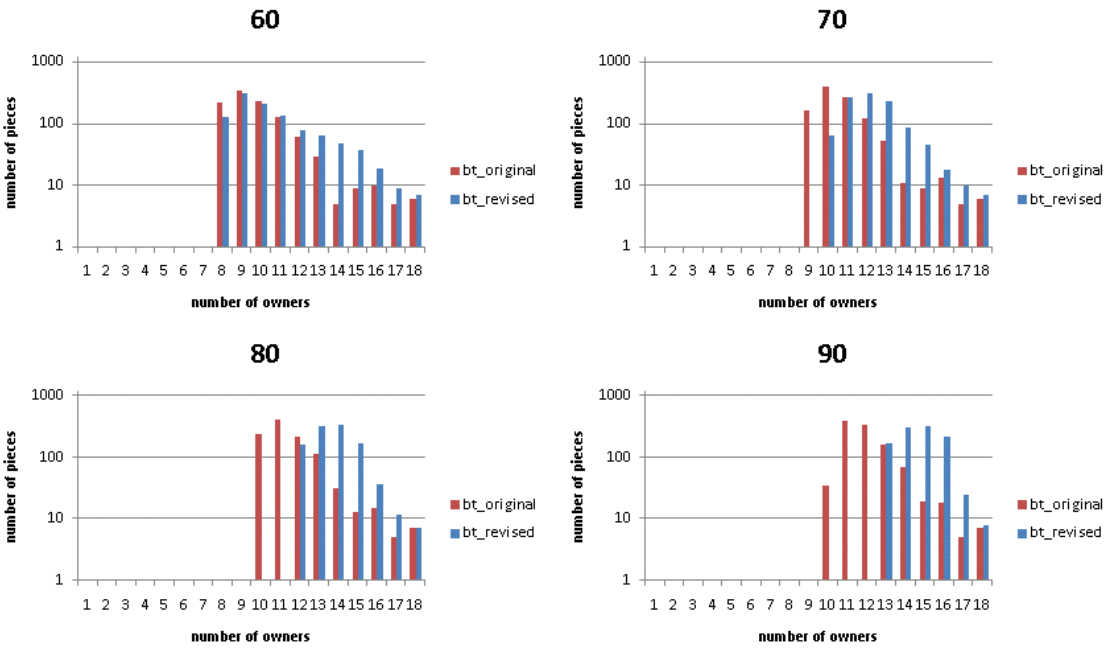
圖表 18：Data Download Rates of Peers with Upload Rates 5KBps



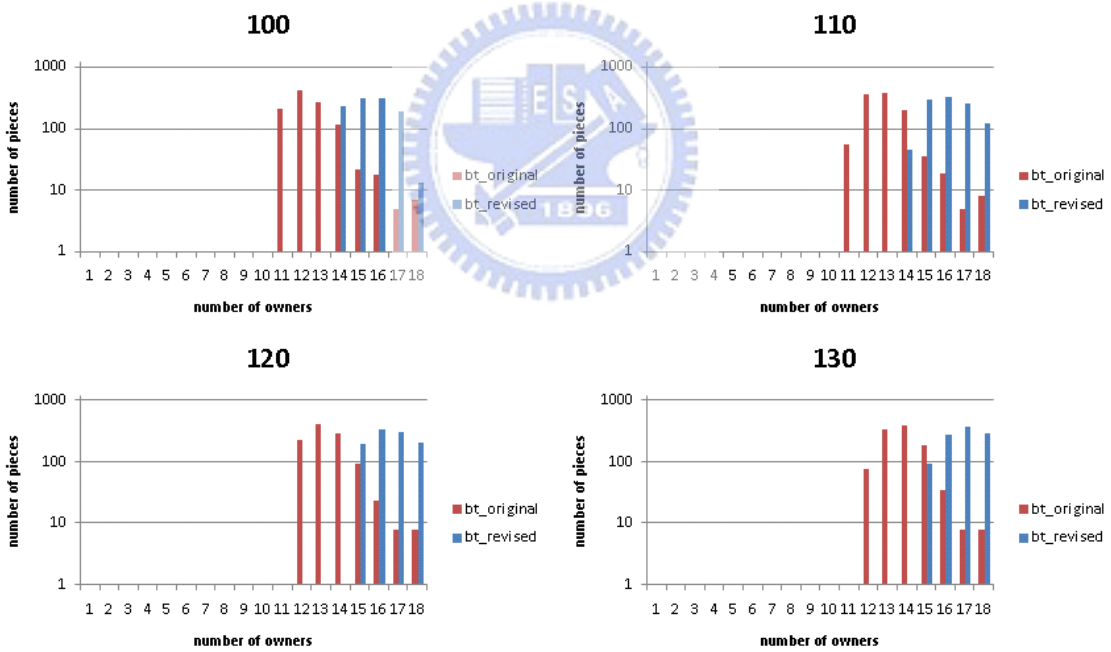
圖表 19：Data Diffusion Rates: Piece Count vs. Owner Count (Start Period, 1 minute – 15 minutes)



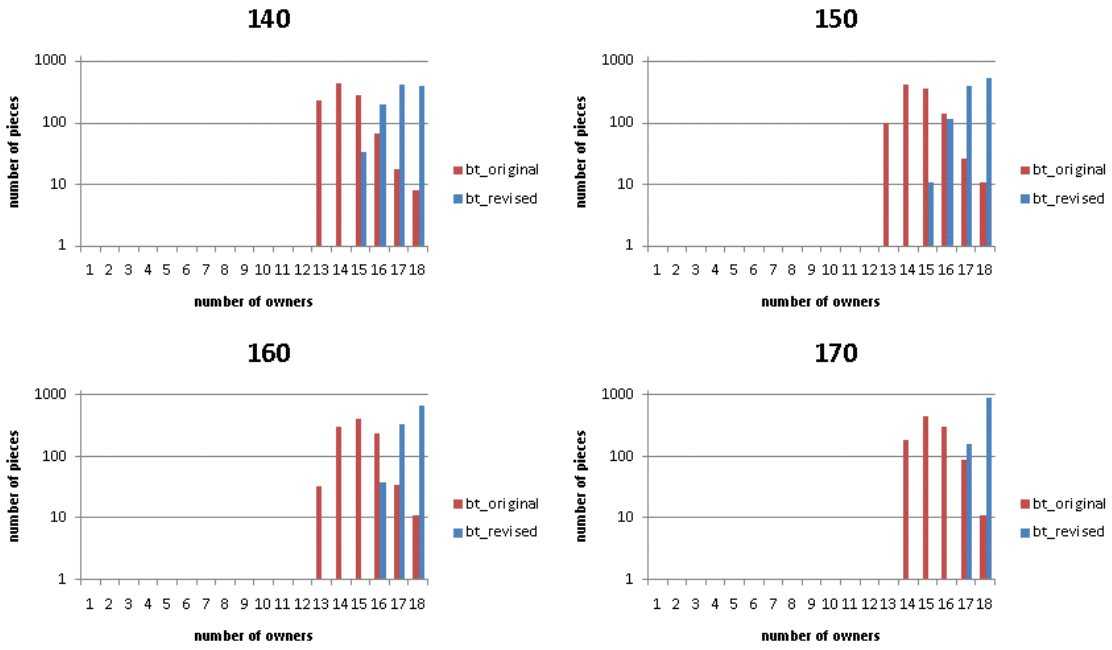
圖表 20：Data Diffusion Rates: Piece Count vs. Owner Count (Early Period, 20 minutes – 50 minutes)



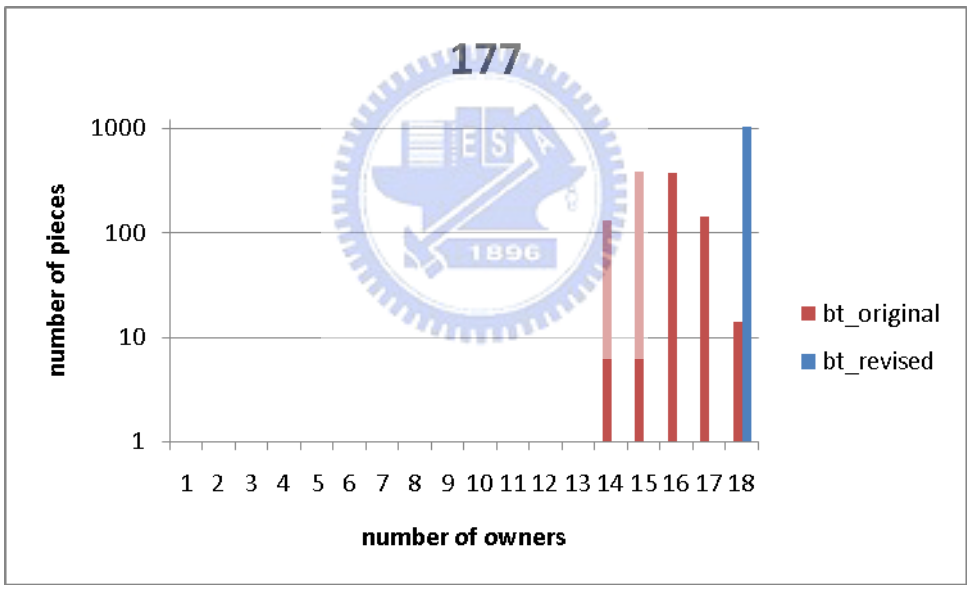
圖表 21：Data Diffusion Rates: Piece Count vs. Owner Count (Middle Period, 60 minutes – 90 minutes)



圖表 22：Data Diffusion Rates: Piece Count vs. Owner Count (Late Period, 100 minutes – 130 minutes)

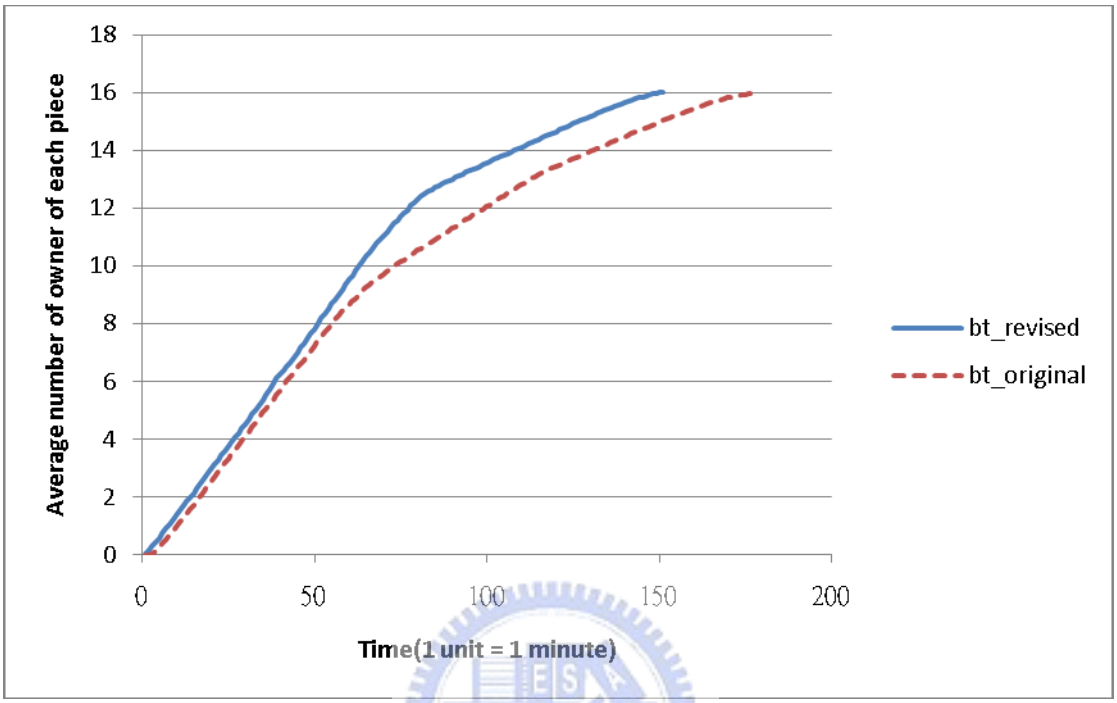


圖表 23：Data Diffusion Rates: Piece Count vs. Owner Count (End Period, >140 minutes)

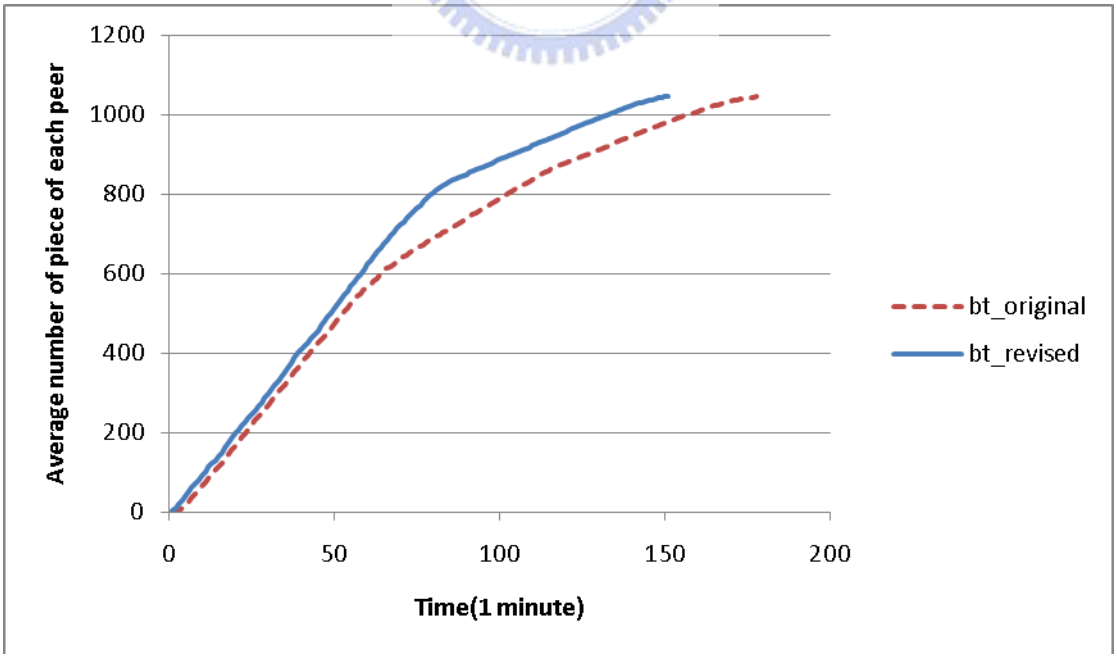


圖表 24：Data Diffusion Rates: Piece Count vs. Owner Count (Finish Time, 177th minutes)

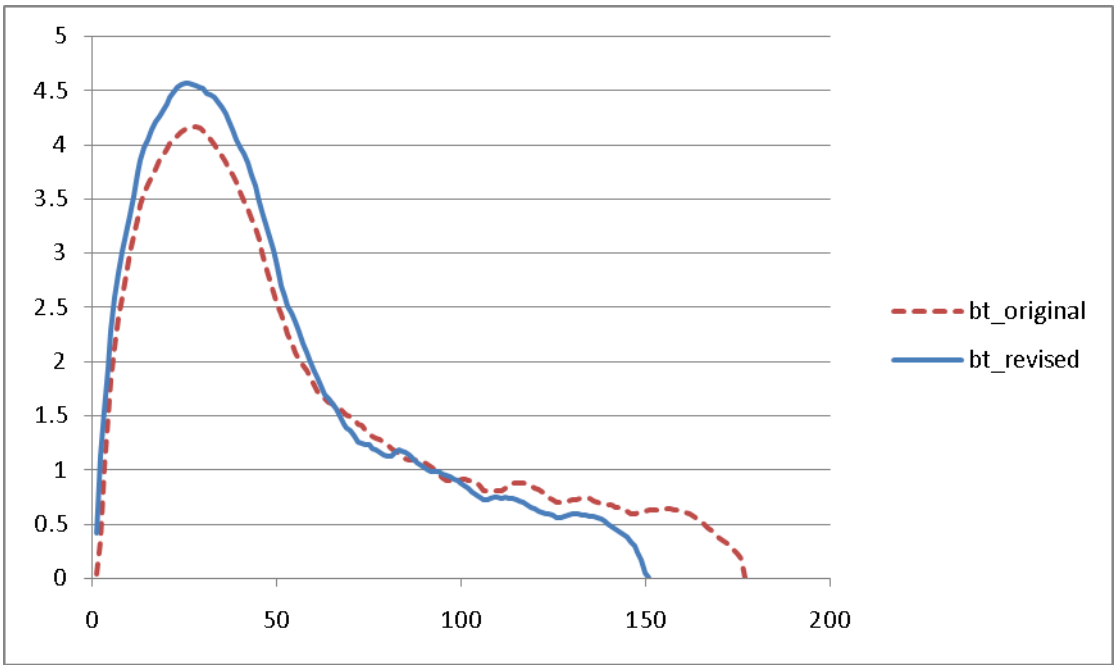
Appendix B (Results of Aug. 8)



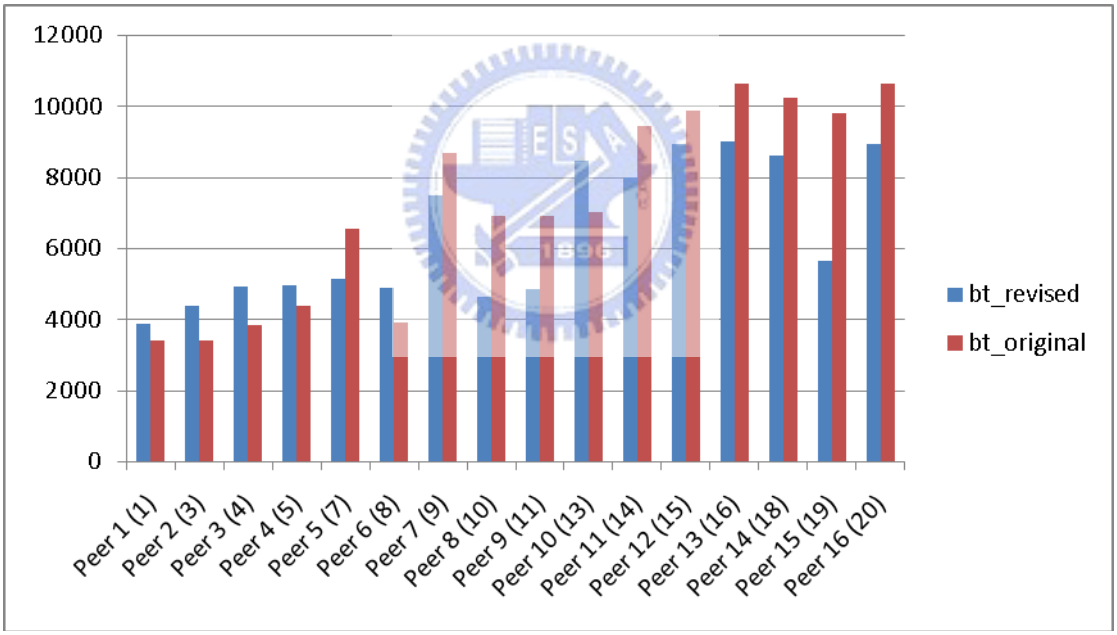
圖表 1：Data Diffusion (Avg. Peer Count / Piece)



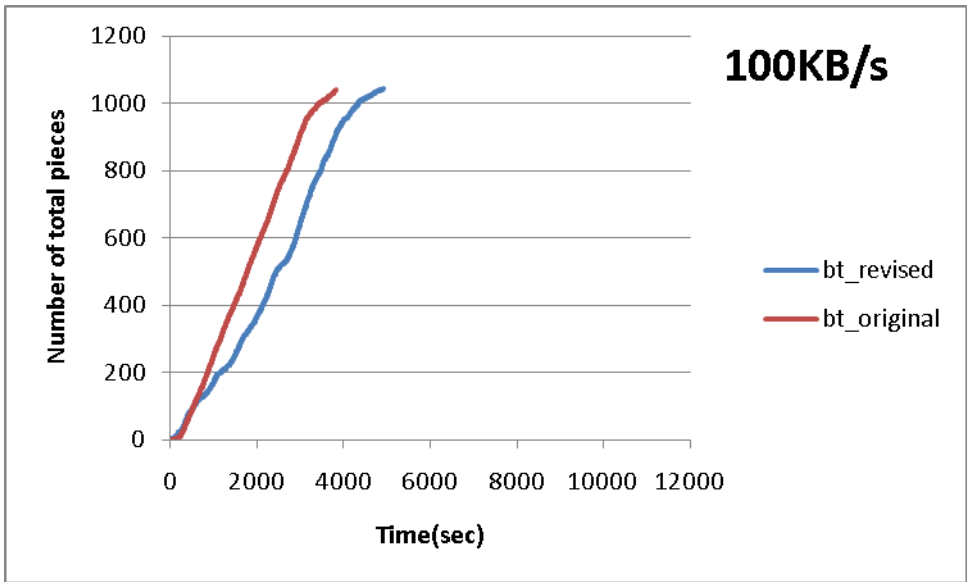
圖表 2：Data Accumulation (Avg. Piece Count / Peer)



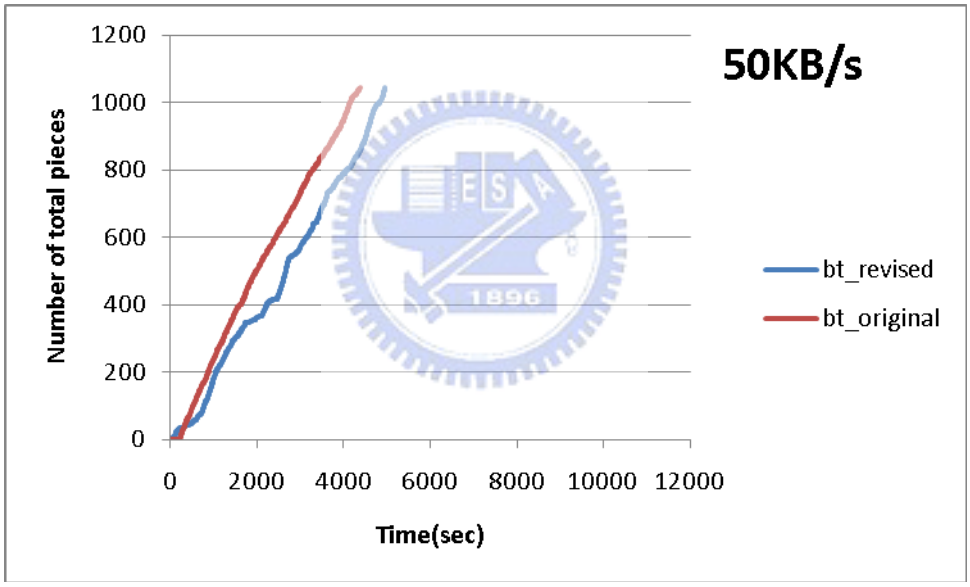
圖表 3：Data Diffusion (Standard Deviation of Owner Count)



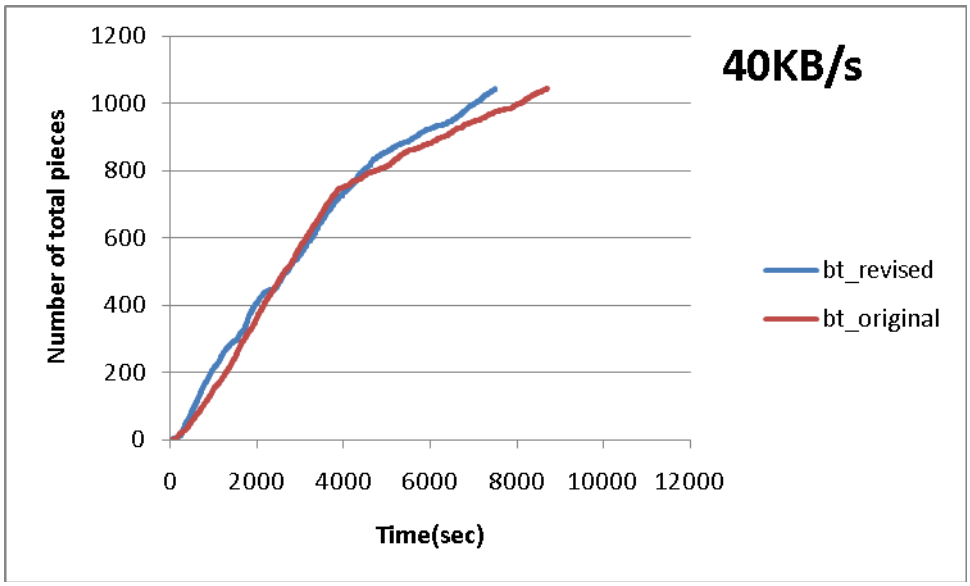
圖表 4：Session Duration(Peer Completion Time)



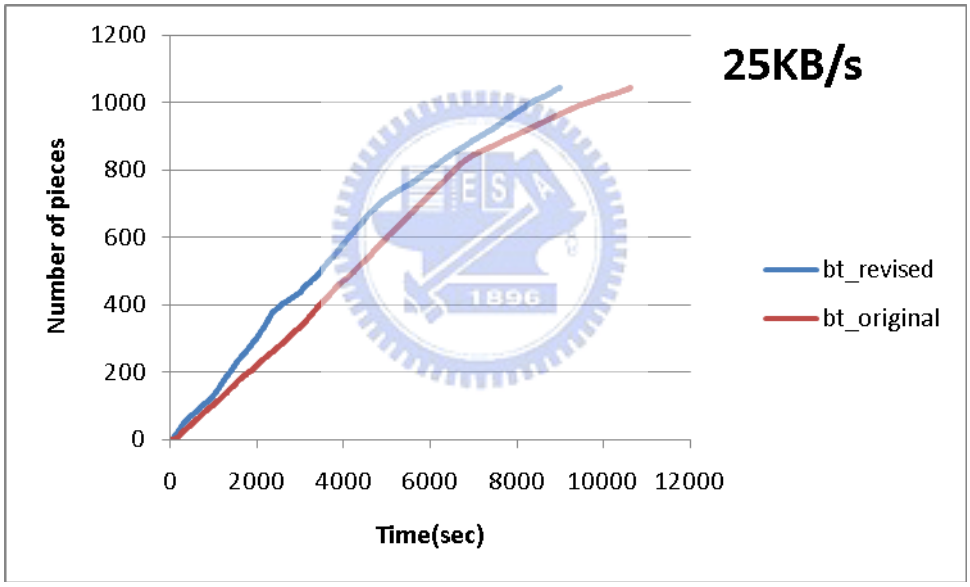
圖表 5：Data Accumulation of Peers with Upload Rates 100KB/s



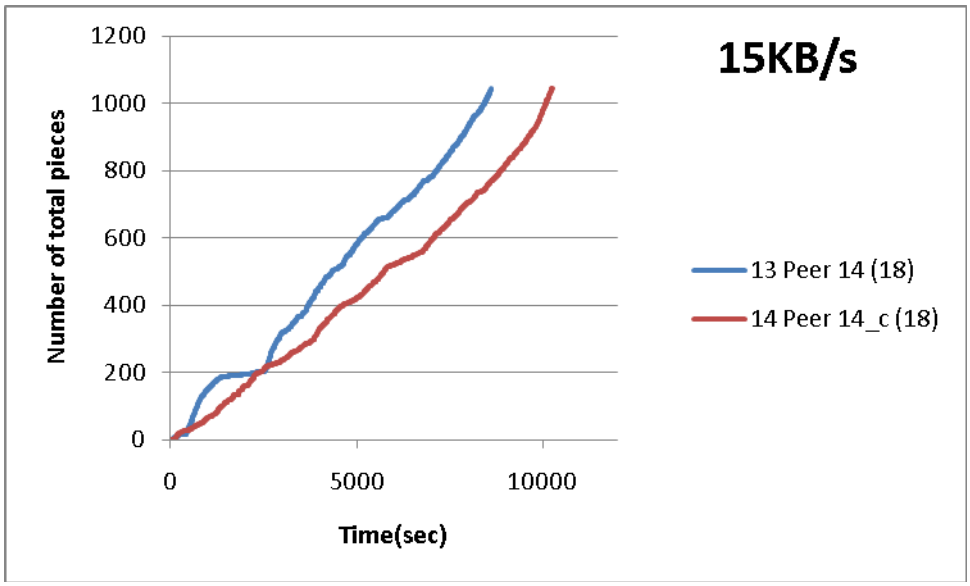
圖表 6：Data Accumulation of Peers with Upload Rates 50KB/s



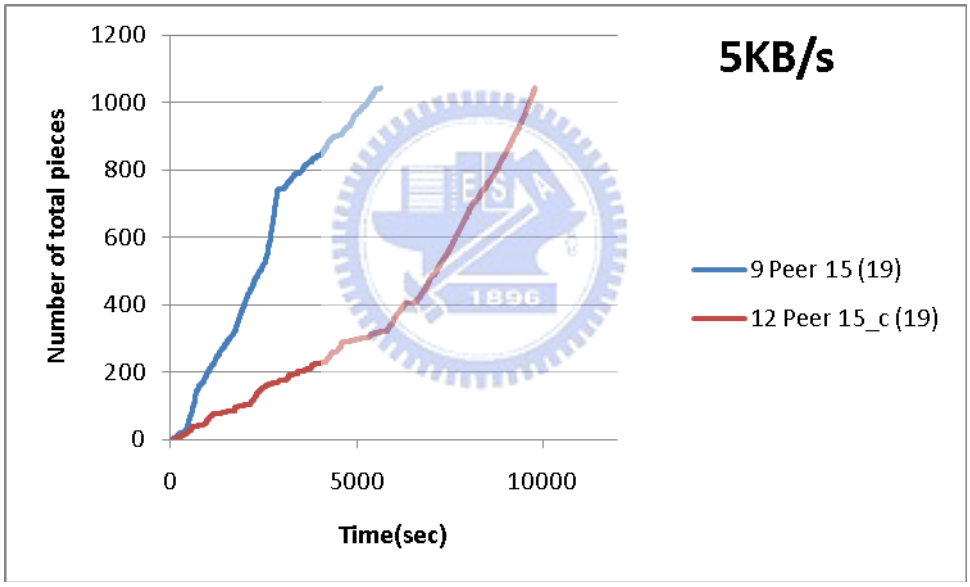
圖表 7：Data Accumulation of Peers with Upload Rates 40KB/s



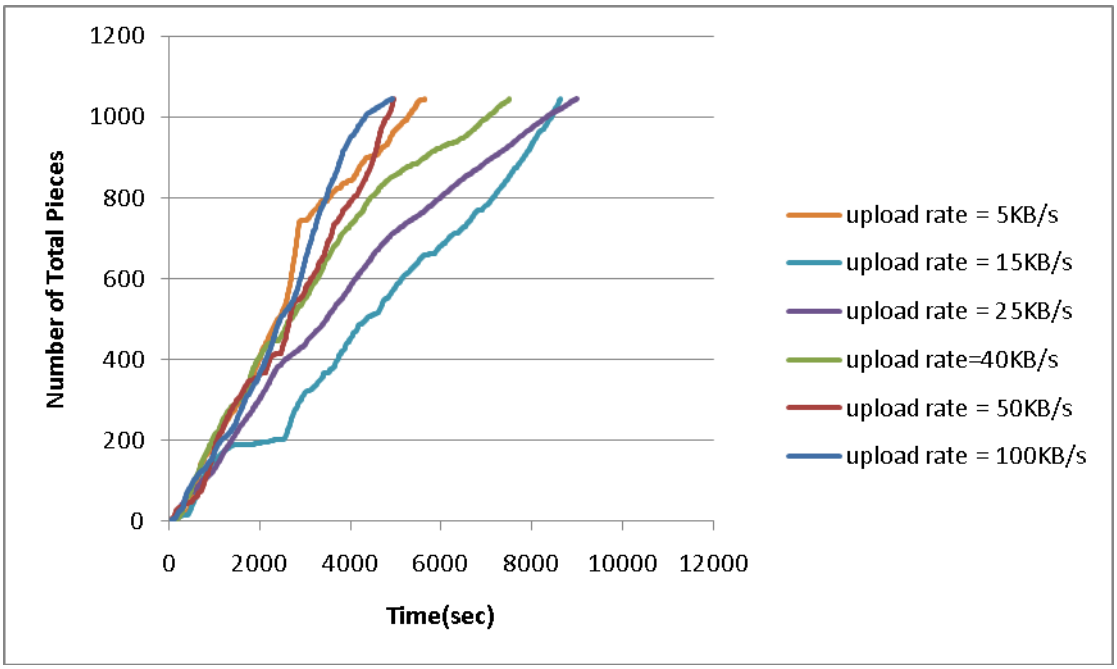
圖表 8：Data Accumulation of Peers with Upload Rates 25KB/s



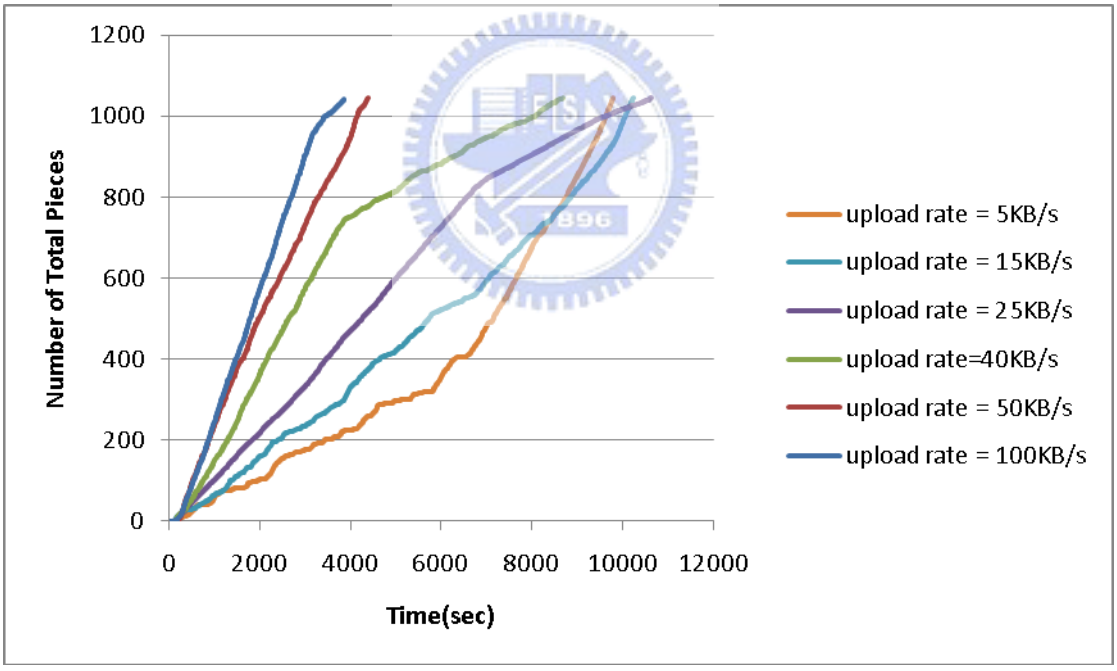
圖表 9：Data Accumulation of Peers with Upload Rates 15KB/s



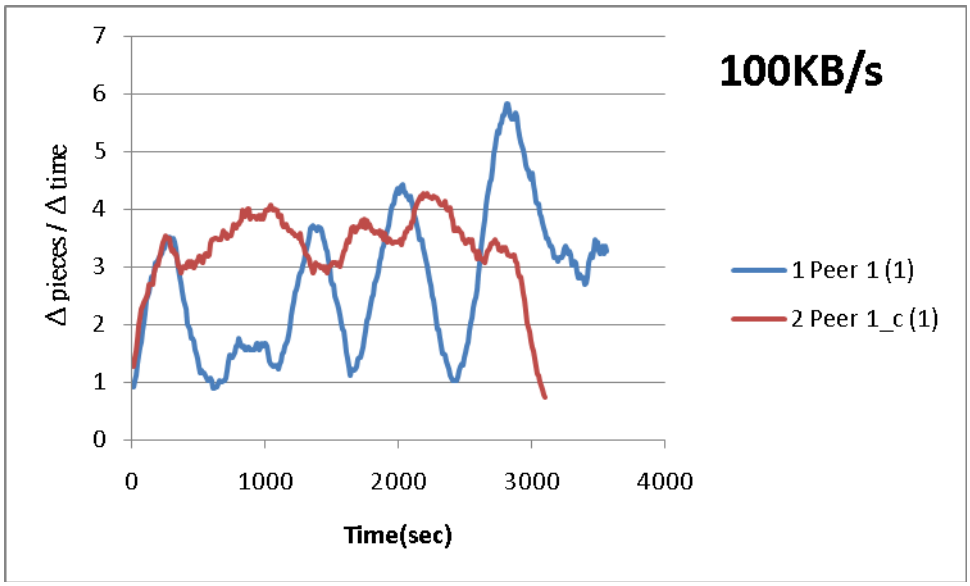
圖表 10：Data Accumulation of Peers with Upload Rates 5KB/s



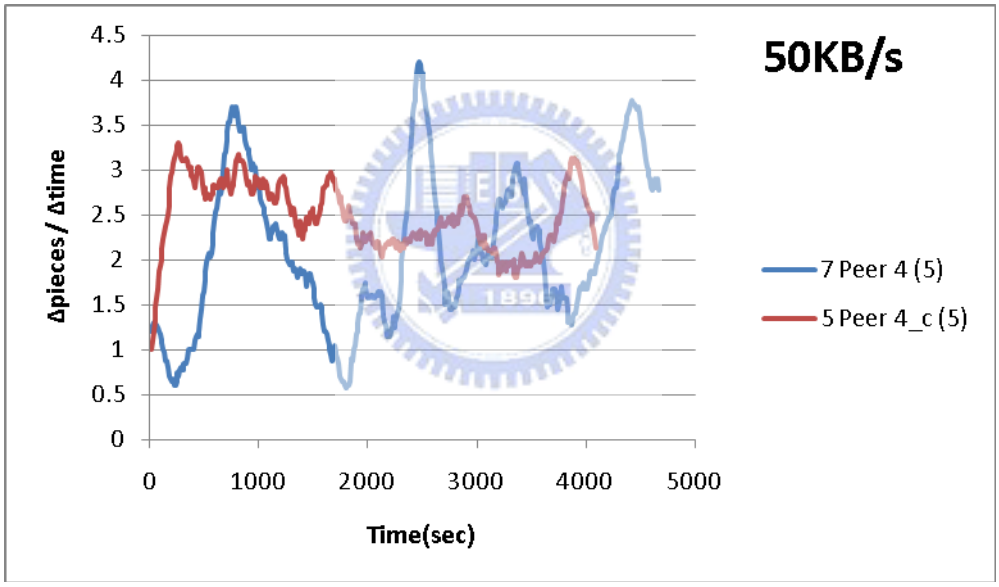
圖表 11：Comparison of Data Accumulation of Peers Using BT*



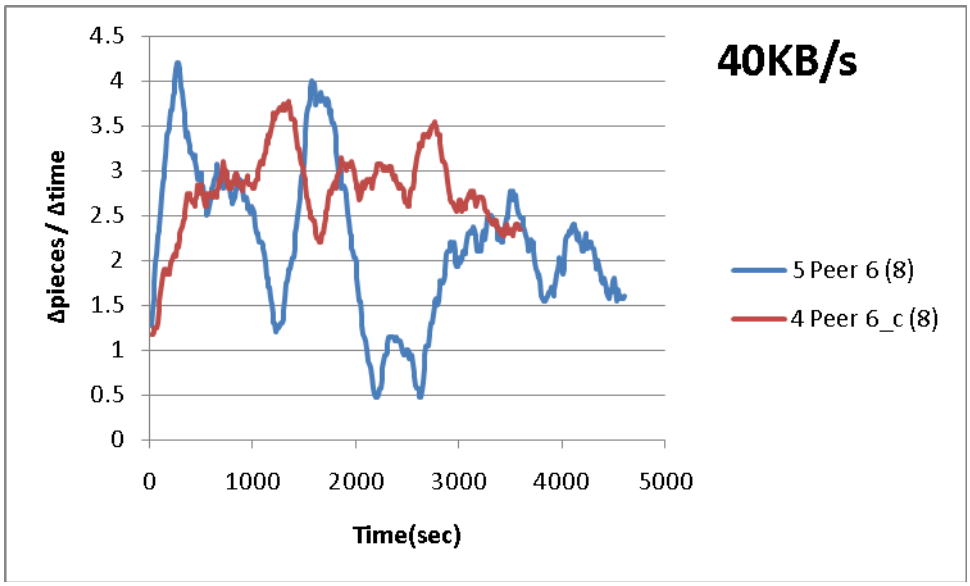
圖表 12：Comparison of Data Accumulation of Peers Using BT



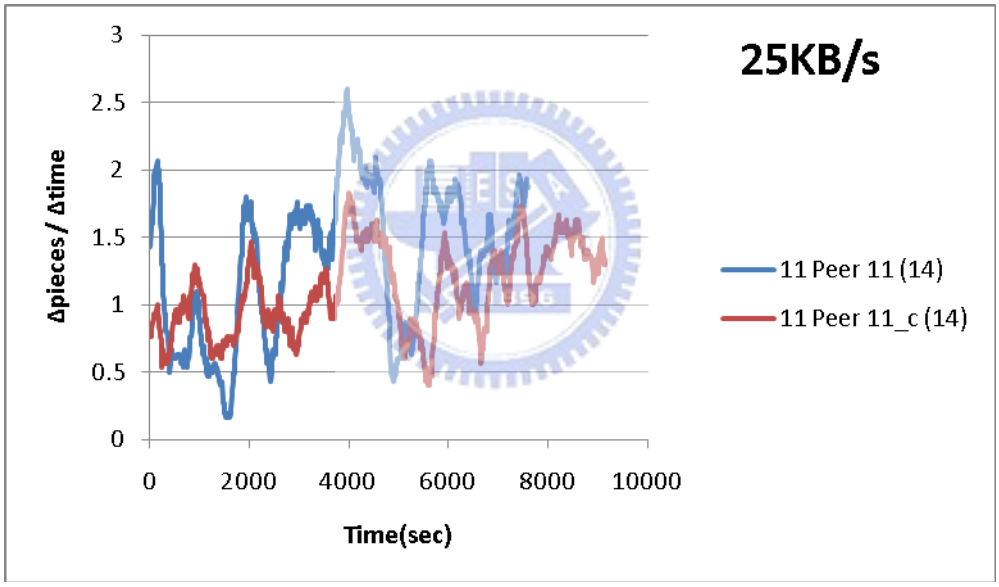
圖表 13：Data Download Rates of Peers with Upload Rates 100KBps



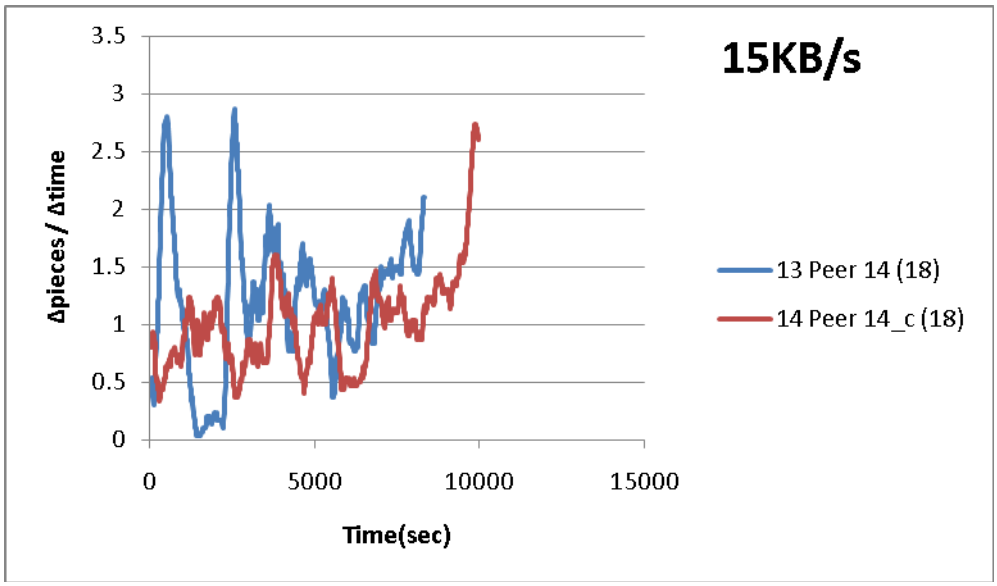
圖表 14：Data Download Rates of Peers with Upload Rates 50KBps



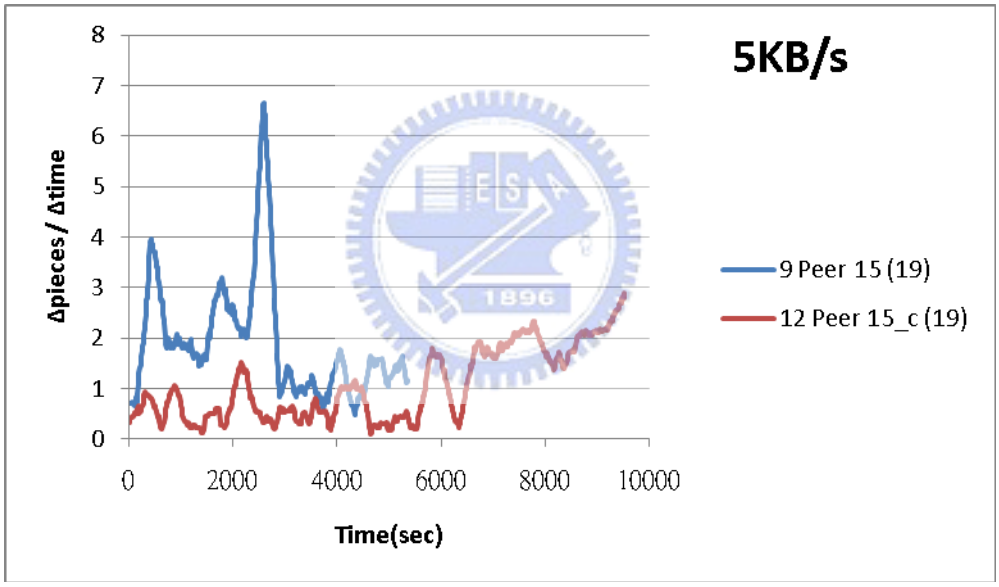
圖表 15：Data Download Rates of Peers with Upload Rates 40KBps



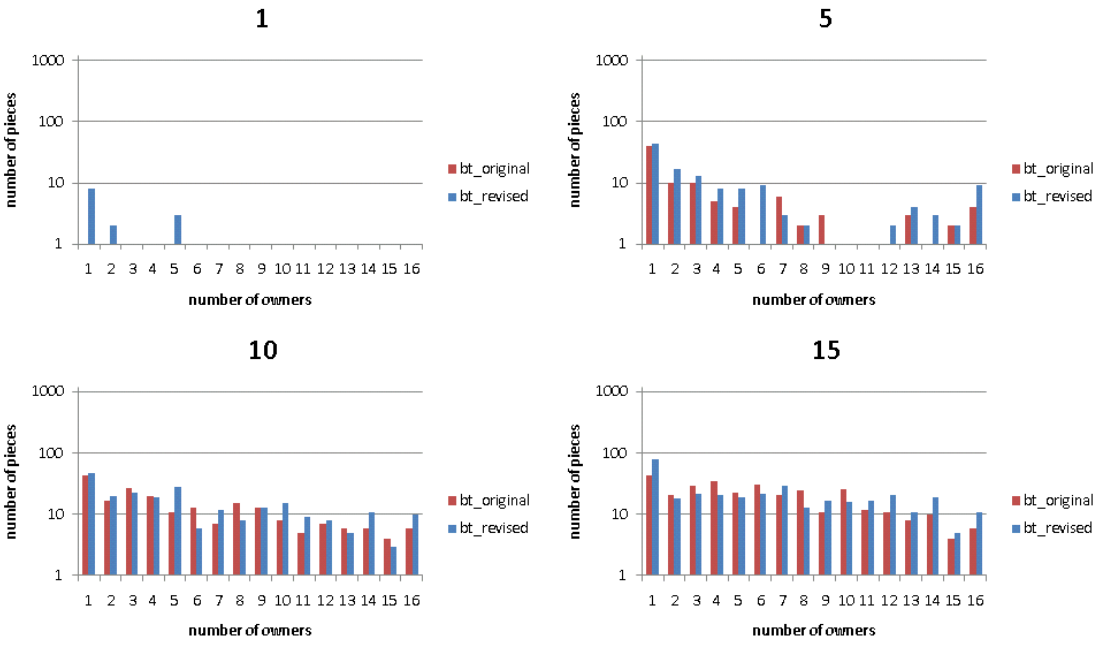
圖表 16：Data Download Rates of Peers with Upload Rates 25KBps



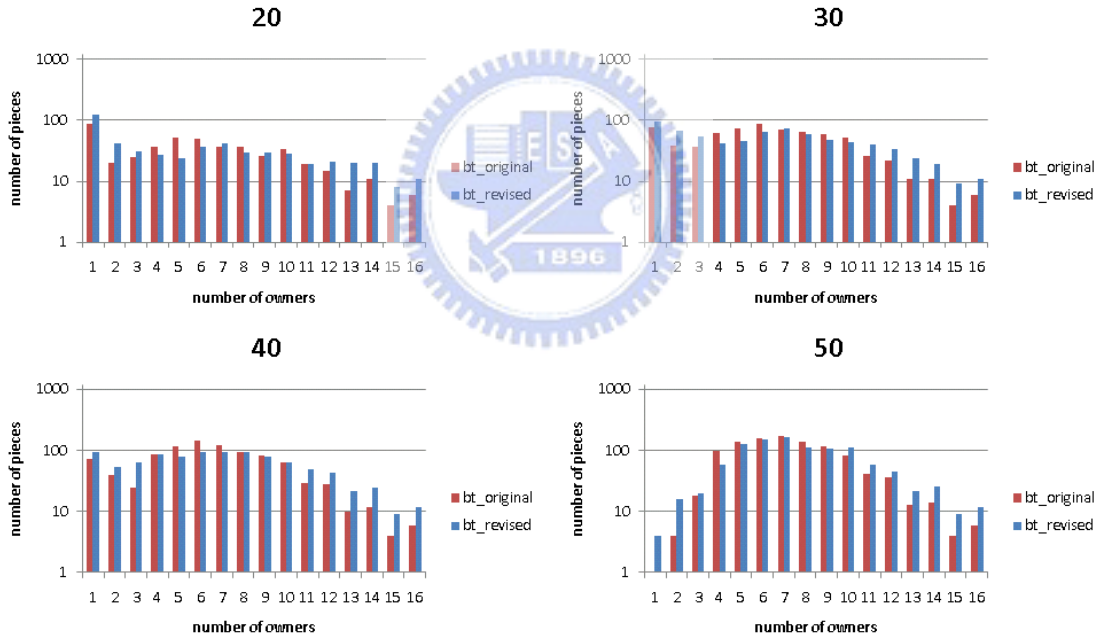
圖表 17：Data Download Rates of Peers with Upload Rates 15KBps



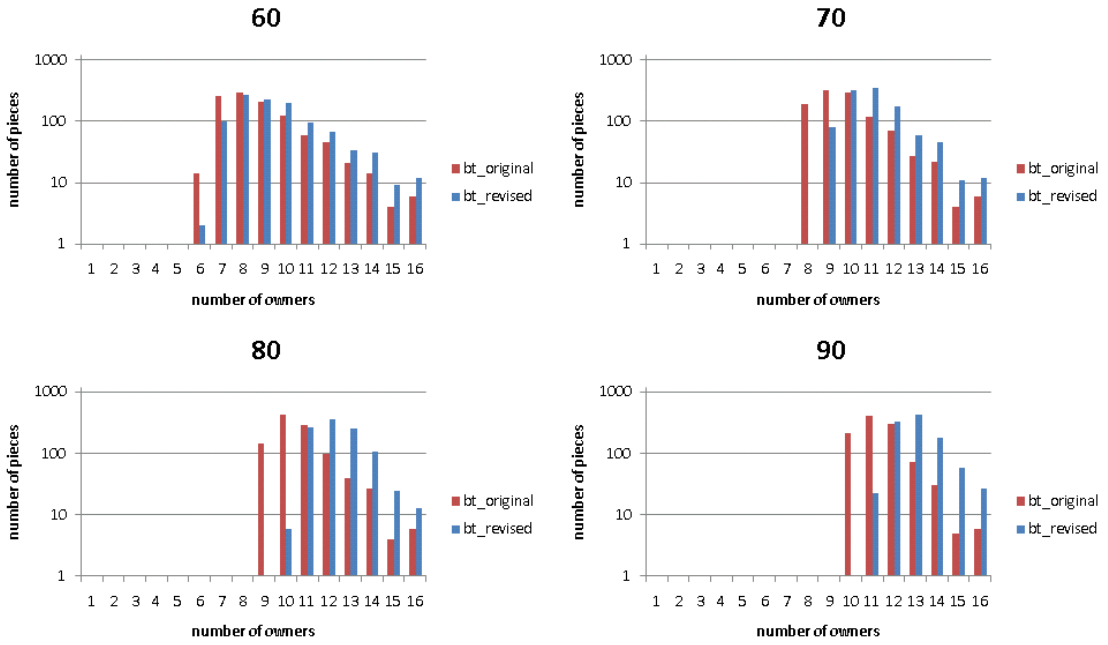
圖表 18：Data Download Rates of Peers with Upload Rates 5KBps



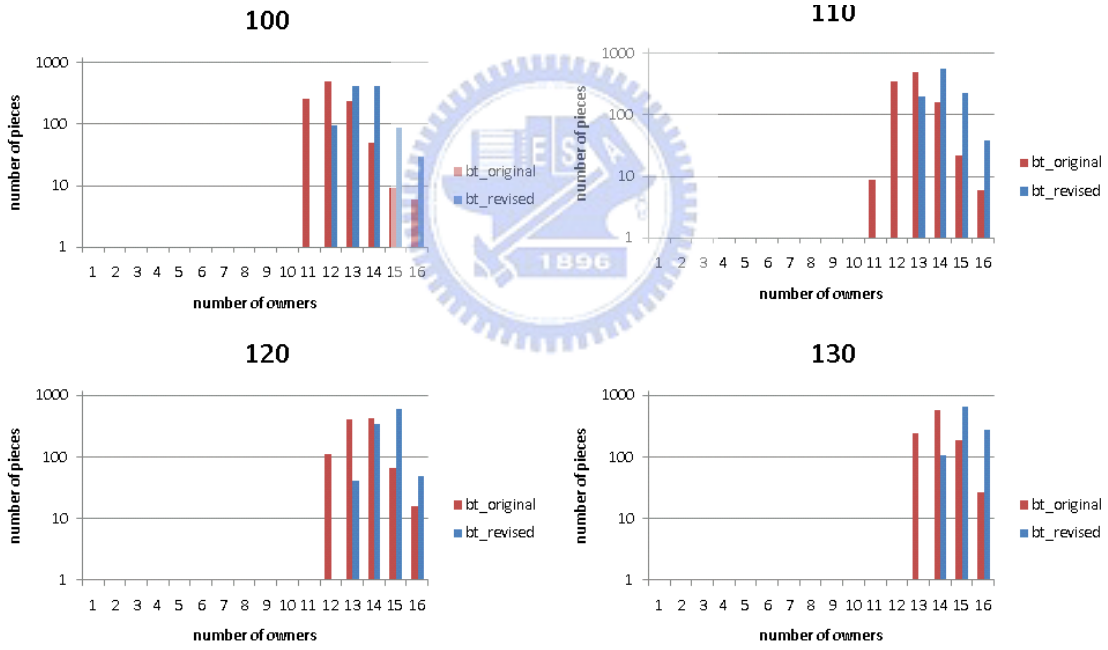
圖表 19：Data Diffusion Rates: Piece Count vs. Peer Count (Start Period, 1 minute – 15 minutes)



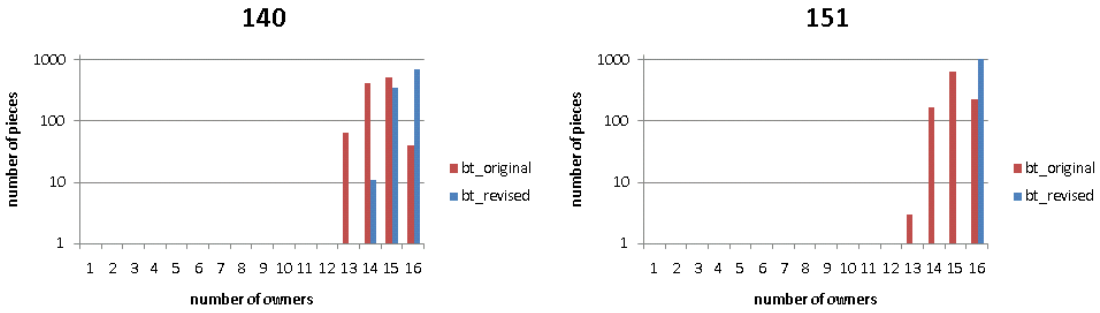
圖表 20：Data Diffusion Rates: Piece Count vs. Peer Count (Early Period, 20 minutes – 50 minutes)



圖表 21：Data Diffusion Rates: Piece Count vs. Peer Count (Middle Period, 60 minutes – 90 minutes)



圖表 22：Data Diffusion Rates: Piece Count vs. Peer Count (Late Period, 100 minutes – 130 minutes)



圖表 23：Data Diffusion Rates: Piece Count vs. Peer Count (End Period, >140 minutes)