國 立 交 通 大 學

網路工程研究所

碩 士 論 文

基於同儕式網路隨選視訊串流的

高強度區塊作種與排程演算法

Robust Seeding and Scheduling Algorithms

for Segmented Peer-to-Peer VoD Streaming

研 究 生：蘇彥霖

指導教授：邵家健 博士

中 華 民 國 九 十 九 年 十 一 月

基於同儕式網路隨選視訊串流的高強度區塊作種與排程演算法

Robust Seeding and Scheduling Algorithms for Segmented Peer-to-Peer VoD Streaming

研 究 生：蘇彥霖　　Student：Yen-Lin Su

指導教授：邵家健 博士　　Advisor：Dr. John Kar-Kin Zao

國 立 交 通 大 學

網 路 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of Master in Computer Science

November 2010

Hsinchu, Taiwan, Republic of China

中華民國九十九年十一月

基於同儕式網路隨選視訊串流的高強度區塊作種與排程演算法

學生：蘇彥霖　　指導教授：邵家健 博士

國立交通大學網路工程研究所 碩士班

摘要

在基於同儕網路的隨選視訊系統中，隨選視訊伺服器端提供影音內容，客戶端則以點對點方式下載並分享影音內容。由於檔案分佈的低同步性，加上快取空間有限，導致擁有的影像片段相異，造成同儕網路的分享與下載效能不彰。另外考量到網路使用花費與網路頻寬限制，隨選視訊系統客戶端必須協助伺服器端作種（Seeding）分享以節省其網路租用成本。如何解決這些問題並增進分享效率是本論文的研究課題。

針對此問題，我們將影片切割成若干個小型的影音區塊（Segment）作為下載與分享使用的單位，並分別製成種子檔案（Torrent）與作種，進而根據影像播放的優先順序，對作種與下載的任務設計出一套排程方法，並預測下載速度與播放緊急程度，達到避免播放影片時產生的中斷情形，提升影片播放品質，並增進同儕間之作種分享效能。

由實驗中我們觀察到下載演算法中緊急指標之低水線越低，對使用者播放中斷情形有較佳之影響，且作種方面以高頻寬配合較多作種個數會表現最佳之結果。

Robust Seeding and Scheduling Algorithms for Segmented Peer-to-Peer VoD Streaming

Student: Yen-Lin Su Advisor: Dr. John Kar-Kin Zao

Institute of Computer Science and Engineering

National Chiao Tung University

ABSTRACT

In P2P VoD system, VoD server supports the video content and clients download and share by P2P. Further, VoD clients have less synchrony, cause them to share weakly. Consider the cost of network and limit on bandwidth, clients must help sharing server to reduce costs. The approach of this thesis will solve these issues.

We separate a film into many small video “segment” as a unit of downloading and seeding, take into torrent files and to be seed. Based on the time priority of playing file, we provide a segment scheduling algorithm, expect enhance smoothness when user watching files, and have outstanding performance of sharing among peers.

In the experiments, we found out when L.U. is low, user experience is better. Seeding algorithm have the best performance when high bandwidth with large seeding size.

誌謝

首先要感謝邵家健老師與工研院資通所劉炳傳先生這段時間以來的指導與諄諄教誨，每每與大家討論問題後總能讓我獲益良多，進而激發更多想法，也真正瞭解何謂作研究，能夠完成這篇論文，要歸功邵老師與劉先生。也十分感謝工研院郭倫嘉經理以及曹孝櫟教授擔任畢業口試委員，不但指出了我在研究的盲點，更給予我許多寶貴的意見，謝謝各位。

接著要感謝實驗室的同學與學弟妹們：嘉錡、星閔、博政、梅瑛、勝焜、子晉、鈞凱、俊瑋、志明等人，在這段日子裡多虧有你（妳）們一起分享各種甘苦，另外也十分感謝偉翔幫我執行各種實驗並製作圖表。

謝謝我的朋友孝謙、承宣、偉程、贊豐、人仲，在我進行研究的期間，帶給我許多想法，也幫助我在模擬時解決不少困難，另外也感謝威凱、曦德、孟雁、晨宏、重翰、盈羽、元德、志鴻、紋德、偉珍、寶合、家駿在這段時間替我打氣，使我在這段人生的低潮期有繼續奮鬥的決心並能樂觀以對。

最後，要感謝一直以來為我操心煩惱，並體諒我、支持著我的家人，讓我永無後顧之憂，專心在研究上，再多的篇幅也道不盡我心中的感謝，謝謝！

目錄

圖目錄

表格目錄

Chapter 1　綜論

1.1　問題陳述

隨著網路頻寬的成長，影音多媒體串流（Media Streaming）的需求可說日益劇增。隨選視訊系統（Video-on-Demand；VoD）[1]的發展是目前網際網路應用服務的主流之一。隨選視訊系統採用主從式架構（Client／Server），將視訊內容存放於遠端視訊伺服器（VoD Server），經網路傳輸後送達客戶端（Client）並播放。而隨選視訊系統需傳輸的視訊內容大、頻寬需求高、服務時間較長，故伺服器需要較高之系統要求，也增加不少網路租用上的成本[2]。

因此有人提出使用點對點同儕式網路（Peer-to-Peer；P2P）[3]技術，不同於主從式架構，使用者（Peer，在同儕網路中亦稱為節點）可同時作為伺服器端與客戶端，同儕網路上節點收到影音串流內容後，可再次將該資料轉送予其他使用者，協助傳輸影音多媒體串流，以減輕影音多媒體提供者在網路與主機能力上之負擔。

在基於同儕網路的隨選視訊系統中，隨選視訊伺服器端提供影音內容，客戶端則以點對點方式下載並分享影音內容。由於客戶端在同一時間欲播放的影像片段不盡相同，加上快取空間有限，導致擁有的影像片段相異，亦不易與其它節點進行分享；每個節點下載影像片段時沒有足夠的檔案提供者，這種檔案分佈的低同步性，造成同儕網路的分享與下載效能不彰。另外又考量到網路使用花費與網路頻寬限制，隨選視訊系統客戶端必須協助伺服器端作種（Seeding）分享以節省其網路租用成本，在下載與作種的任務分配管理須有所平衡。如何解決這些問題並增進分享效率是本論文的研究課題。

1.2　研究方法

針對這個問題，我們將影片切割成若干個小型的影音區塊（Segment），作為下載與分享使用的單位，並分別製成種子檔案（Torrent）與作種，進而根據影像播放的優先順

序，對下載與作種的任務設計出一套排程方法。期待增進同儕間之作種分享效能，以協助伺服器端，達到形同多個伺服器的分享能力，使整體網路能負荷大量的使用者觀看而不失其影片播放品質，避免使用者播放影音時可能產生的中斷情形（Playback Interruption）。

1.3 論文大綱

本論文將分成五個章節：第二章介紹基本的背景知識及相關文獻，包括BitTorrent[4][5][6]、BASS[7]、CoolStreaming[8][9]、BiToS[10]等。第三章說明本研究在如何以BitTorrent為基礎的下載系統中，針對隨選視訊服務的下載及作種任務設計出排程方法。第四章為本研究所設計的模擬環境與結果。第五章則總結本研究的貢獻與未來可改進的方向。

Chapter 2 背景知識

同儕式網路影音串流傳輸，是目前最受重視的研究方向。因為同儕式網路的機制能夠有效地節省伺服器端頻寬與計算能力，非常適合在網際網路上提供影音服務並節省營運成本。BitTorrent 是一種基於同儕網路的檔案傳輸分享協定，因為特殊的獎勵（Incentive）機制設計，在網際網路上被廣大的使用者下載做為檔案分享的媒介，而且也證明了其同儕式網路之優異性。對於流量日益劇增的網路影音串流，有許多類 BitTorrent 的同儕網路串流設計（如 BASS、CoolStreaming、BiToS）；另針對隨選視訊系統亦有許多相關研究。

2.1 BitTorrent

BitTorrent 是一個熱門的 P2P 的檔案分享架構，於 2001 年由 Bram Cohen 發明，它將檔案提供者發佈出來分享的檔案切成許多的檔案片段（Piece），下載者下載部份片段並互相分享彼此的片段，將上傳的負荷分配給下載者。

根據 BitTorrent 協議，檔案發佈者會根據要分享的檔案產生一個種子檔案文件，也簡稱為 Torrent；Torrent 檔描述了原始檔案的一些特性，以及想要下載這個檔案的同儕們必需知道的基本資訊與同儕列表記錄者（Tracker）資訊，Tracker 是一個集中式的伺服器，並不負責實際檔案的傳輸，而是幫助這些要下載檔案的同儕可以找到彼此，主要工作內容是負責紀錄有哪些同儕正在下載或分享該檔案，而每個同儕也會定時向 Tracker 通知自己的最新下載狀況。

當使用者想要透過 BitTorrent 網路去下載某個檔案的時候，他必須要先取得此檔案的種子檔，而種子檔一般可透過 HTTP 協定向網頁伺服器取得，接著使用者根據種子檔提供的 Tracker 的 IP 位址連到 Tracker 之後，Tracker 回傳給下載者同樣在下載這個檔案的同儕列表清單。下載者即根據這個清單直接與其它同儕交換檔案片段，進而下載完整檔案。

2.1.1 Piece Selection

由於同儕常與多個檔案提供者請求下載服務，故會產生向哪位同儕請求下載哪一個片段會較有效率的問題，因此在 BitTorrent 的架構中，片段選擇（Piece Selection）就成為一個重要的議題。片段選擇的主要目的就是希望儘快將不同的片段散佈到不同的同儕中，以增加同儕彼此之間的分享效率，主要的片段選擇法可以分為下面四種：（1）Strict Priority（2）Rarest First（3）Random First Piece（4）Endgame Mode。

2.1.2 Peer Selection

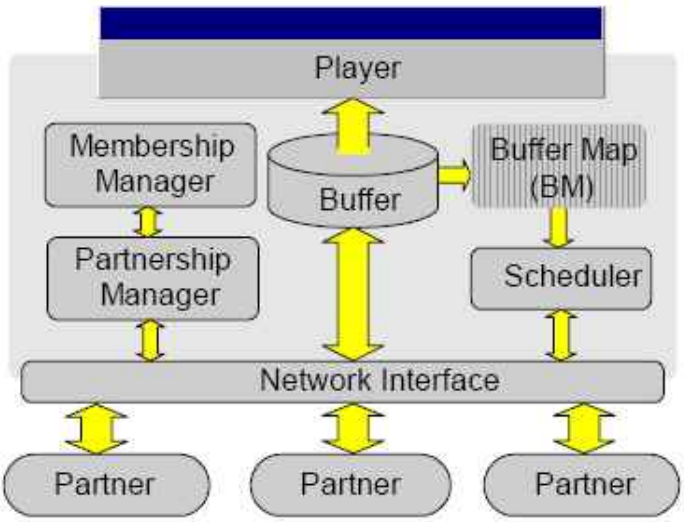
由於BitTorrent技術不是集中式的資源分配，每一個同儕都有責任讓自己的下載速率最大化。當同儕下載時，會根據對方同儕提供我們的下傳速度給予適當的回報，對於有貢獻的同儕則對它提供上傳服務，對於沒有貢獻的同儕就給予阻塞（Choking）。Choking是暫時拒絕上傳給檔案對方，但仍可以下載，而當Choking暫停時，連線不需要重新建立。Choking不是BitTorrent的主要技術，但它對於BitTorrent整體的效能表現來講是很重要的機制。它讓所有可用資源的使用分享最大化，並且提供每個人合理的下載速度，主要分為：（1）Pareto Efficiency（Tit-For-Tat）[1]（2）Choking Algorithm（3）Optimistic Unchoking（4）Anti-Snubbing（5）Upload Only。

2.2 CoolStreaming／DONet

在 2004 年，由香港科技大學的Bo Li團隊發表，並實作了DONet（Data-driven Overlay Network）這個架構成為CoolStreaming。CoolStreaming使用Gossip協定來散播同儕列表清單，類似於BitTorrent的點對點的傳輸協定，可在用戶（Partner）之間互相傳送多媒體資料封包，並使用隨機的Peer Selection（Tit-For-Tat），在傳遞多媒體內容時也使用「混合式pull and push」策略，系統主要包含三種模組，分別為Membership Manger、Partnership

[1] Tit-For-Tat 的行為模式是先將過 20 秒內上傳給自己的速度大於 256 Bytes/sec 的該 Peers，根據其上傳速度由快至慢做排序後並依序提供它們下載自己的檔案片段。

Manager與Stream Manager，他們彼此分工，使得系統更加易於佈建、更有效率，也更加強韌與可靠。由於CoolStreaming的成功，使得後期出現的PPLive[11]、PPStream[12]等P2P視訊串流技術也大多沿用了CoolStreaming的運作模式。圖 1 為DONet的系統示意圖。



A generic system diagram for a DONet node.

圖 1：DONet 系統圖

2.3 BASS

BitTorrent Assisted Streaming System(BASS)，為設計於大規模的隨選視訊系統服務，對 BitTorrent 作了少許的修改，並加上外部的多媒體串流伺服器（Media Server），使點對點傳播技術融合主從式架構(如圖 2)，不僅降低多媒體串流伺服器的頻寬負擔，亦減少了串流播放的等待時間。

由於需仰賴串流「依序」地傳送才能正常播放影音，因此 BitTorrent 的最稀有優先策略（Rarest First）不適合作為串流傳播之用，為處理這個問題，BASS 對 BitTorrent 做了些許修改：在下載「目前播放片段點」前不會下載任何資料。

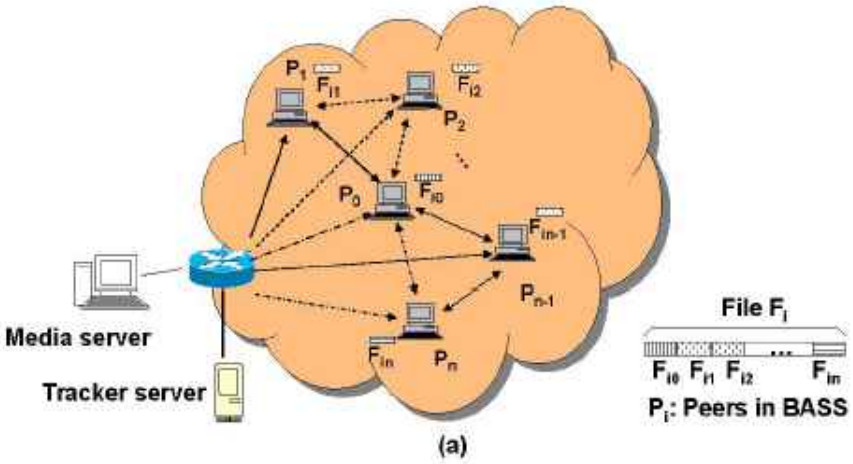


圖 2：BASS 系統架構圖

另外當綜合了多媒體串流伺服器與同儕的連線時，為使 BASS 能依序下載串流的片段，BASS 提供如下圖 3 的 Client Model 運作方式：首先，同儕們先各自下載所需的片段並存於本地儲存設備直到這些片段被使用；再來，為使同儕們能「依序」地接收影片串流片段，同儕們會根據以下三種狀況來「放棄」從多媒體串流伺服器下載影音串流片段：

I. 同儕們已從 BitTorrent 同儕網路部分下載該影音片段，此時略過多媒體串流伺服器。

II. 「正在」從同儕網路中下載片段時，不向多媒體串流伺服器要求串流傳送。

III. 在預計播放的時間抵達前結束收看，此時後段的串流也不需向多媒體串流伺服器接收。

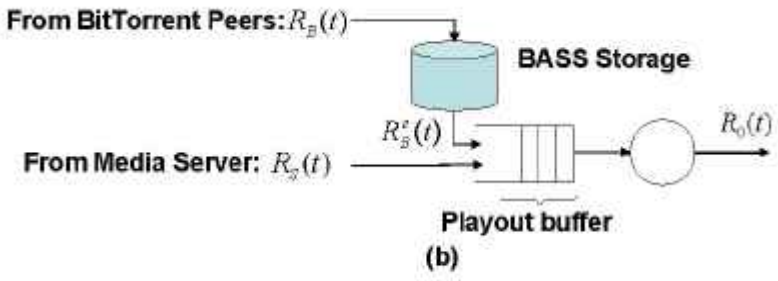


圖 3：BASS Client 模組圖

2.4 BiToS

BitTorrent Streaming（BiToS），屬於 BitTorrent 的一種擴充網路協定，其特色在於 BiToS 為支援串流設計了專用的片段選擇策略。由於 BitTorrent 的 Rarest First 機制並不像影音串流技術需考慮到時間是否過期，故 BiToS 對時間敏感度有較佳之考量，其想法是將需要下載的檔案片段分成 Received Pieces、High Priority Set、Remaining Pieces Set 三個階段，並透過可動態調整的機率變數 p 來控制下載 High Priority Set、Remaining Pieces Set 的比例，以影響其運作策略。

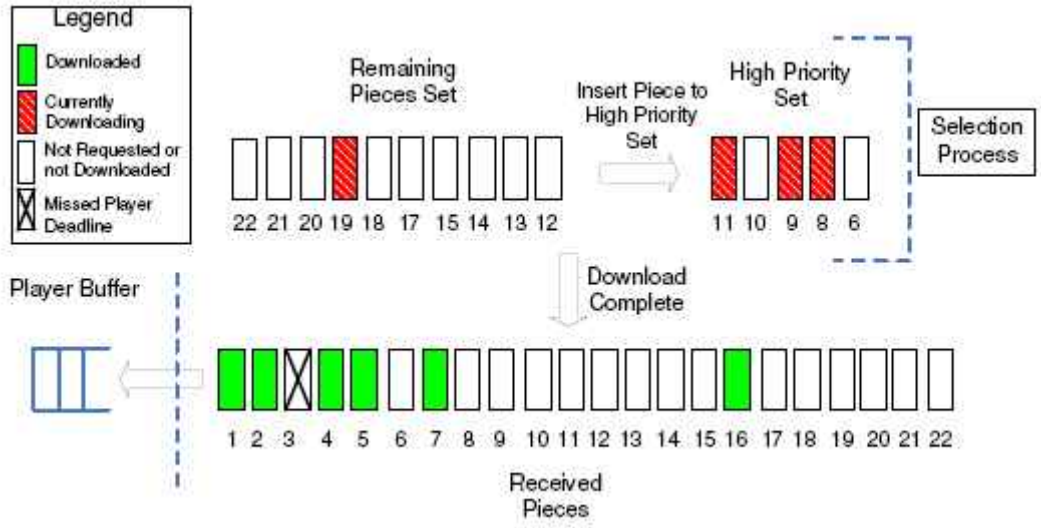


圖 4：BiToS 替 BitTorrent 支援串流之示意圖

2.5 Cache Replacement in Peer-Assisted VoD Systems

運作在同儕網路下的隨選視訊系統，往往需要一塊快取空間（Cache），供下載完成後預備播放與播放後分享他人之用。此篇[13]即探討何種快取替換演算法為較佳之選擇，經過數學推算出最佳化之演算法，與最簡單之基本演算法如LRU、LFU比較後，雖然最佳化演算法的效能較佳，但大多數狀況中仍推薦使用之演算法越簡單越好。

Chapter 3 區塊式作種與排程演算法

3.1 系統分析與解決問題之方案

由於 BitTorrent 是針對檔案下載而生的產物，故在此我們利用 BitTorrent 來負責下載隨選視訊系統提供的影音封包，但由於 BitTorrent 要求整份檔案都下載完成才能執行或播放檔案，這與目標要求即時播放的隨選視訊系統不符。

為消弭 BitTorrent 與隨選視訊系統在使用上的差異，本論文選擇將原始影音檔案切割成若干個小型的影音區塊（Segment），作為下載與分享使用的單位，再分別製成 BitTorrent 下載所需的種子檔，接著透過 BitTorrent 的下載方式來達到點對點傳輸。在下載過程中，為了讓使用者能順暢地從頭收看影片，必須考慮到下載影音區塊的先後順序。再者，由於每位使用者正在觀看與下載的影片檔案不盡相同，加上快取空間有限，不僅導致擁有的影像片段相異，亦不易與其它節點進行分享。且考量到網路使用花費與網路頻寬限制，隨選視訊系統客戶端必須協助伺服器作種分享以節省其網路租用成本。下載與作種的任務分配管理也須有所平衡。

故本論文提出 BitTorrent 在隨選視訊系統客戶端的影音區塊作種排程演算法與下載排程演算法，期望將 BitTorrent 的點對點分享效能延伸至隨選視訊系統的影音傳輸與分享上，增進同儕間之作種分享效能，達到客戶端充份協助伺服器端分享的能力（Load Sharing）；並提升影片播放品質，降低播放影片時產生的中斷（Playback Interruption）情形。

3.2 基於隨選視訊系統的點對點設計架構

本論文先針對隨選視訊之客戶端設計了緩衝區（Buffer）架構。如圖 5，客戶端中的每個同儕則在緩衝區中對其角色功能作不同的安排，分別維護下載與作種的狀況。下載區間（Downloading Segment）的主要任務是負責下載未來將播放的檔案區塊；就

BitTorrent 的原理來說，它同時也正在上傳已下載的部分檔案片段（Pieces），一旦下載完成即進行轉移動作。由於下載完的影音區塊必須依序播放，使得緩衝區仍須額外提供播放用途，故在此額外分出播放區間（Playback Segment）與預載區間（Preloaded Segment），分別處理正在播放及待播放的影音區塊；最後播放完畢的影音區塊則統一交由作種區間（Seeding Segment）處理。

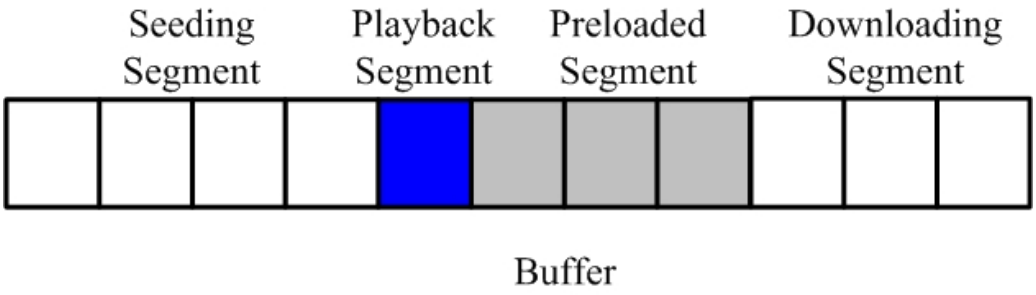


圖 5：VoD Client 的 Buffer 分配示意圖

3.3 下載區塊排程演算法

在此主要目的在於利用本論文的下載排程演算法避免播放中斷。但有幾個決定性的假設前提：

1. 每個區塊的檔案大小不盡相同，最大與最小可能相差 1～3 倍，因此下載時間也有所差異。
2. 第二，區塊下載速度必須大於其播放位元率（Bit Rate）；在此前提下，每個區塊才有機會在播放前完成下載動作，否則下載速度過慢仍無濟於事。換句話說：平均每個區塊距離下載完成的時間必須小於它距離預期播放的時間。
3. 由於客戶端的播放具連續性（按順序播放）的特性，這些連續播放的區塊被趨近於同一批的鄰居同儕所維護與服務，也因此同一客戶端節點相臨區塊編號的下載速度可被認為十分相近。
4. 除了鄰居差異的影響外，我們認為時間也是一項影響要素，也就是說：隨著時間流逝，某個區塊上一次或是上上次所測出的下載速度也可提供參考；且因為鄰居相同，故前幾次測量之下載速度，其參考價值比相鄰編號的區塊下載速度更高。

基於以上前提，本論文制定一些變數幫助了解：

T：每個影音區塊的播放時間

P：在一個播放時間內要檢查是否開始下載的次數

B：檢查是否開始下載的區塊數

k：第 k 個區塊（$1 \le k \le N$）

W_k：第 k 個區塊的下載速度

Q：取出下載速度歷史記錄的時間範圍

R：取出下載速度歷史記錄的區塊數

假設每個區塊的播放時間皆為 T 秒，我們希望在這播放時間內的 P 個時段，也就是系統每隔 $\frac{T}{P-1}$ 秒都能檢查是否有區塊必須開始下載；且每次從 B 個尚未下載的區塊中挑選，最多只能挑選一個開始下載。

由於區塊編號相臨時，下載速度亦相近；在挑選下載區塊時，我們以加權移動平均法（Weighted Moving Average）去推測該區塊開始下載後的下載速度 W，再評估是否該開始下載。加權移動平均法的基本公式如下：

$$W_k = \sum_{i=1}^{n} \frac{\alpha_i W_i}{\alpha_i}$$

其中 W_k 是區塊編號第 k 個的下載速度，而 α_i 是計算範圍內第 i 個區塊下載速度所占的權重。我們的想法是在每次檢查是否有區塊可下載時，取出之前記錄的若干個區塊的下載速度 W，包含下載中或剛下載完畢的區塊，我們以這段評估範圍定為 R。隨著時間流逝，某個區塊的下載速度歷史記錄也是可提供參考的，只是該權重 α 仍因時間流逝而漸漸降低，我們以這段可評估的時間範圍定為 Q。

如圖 6，我們可以看到右邊第 k 個區塊（S_k）正準備計算出預期的下載速度 W_k，x 軸為區塊之編號，其中也包含了下載、預載與播放區塊區間的範圍，y 軸則是代表每次

檢查下載的時間，每次時間間隔如前文提到為 $\frac{T}{P-1}$ 秒，因此每 $\frac{T}{P-1}$ 秒我們也得到並記錄這些下載中區塊的下載速度。

舉例來說：若$B=3$，我們取 4 份時間（$Q=4$，$t\sim t-3$），與最近 5 份區塊的資料（$R=5$），下載區間恰有 3 個區塊正在下載時，則預載區間會找最近的 2 個區塊的速度資料，故理論上我們可以得出一份最多 20 份下載速度的資料表格。但因這些區塊下載狀況不一，剛開始下載的區塊所獲得的速度資料必然較少；且預載區間的區塊因為已下載完成，故更新之下載速度必為 0，故不計算其權重。以此推論，實際能測得之下載速度資料可能如橘色部分所示。在此為易於說明，將 W 與α以 $W_{編號，時間}$、$\alpha_{編號，時間}$表示。

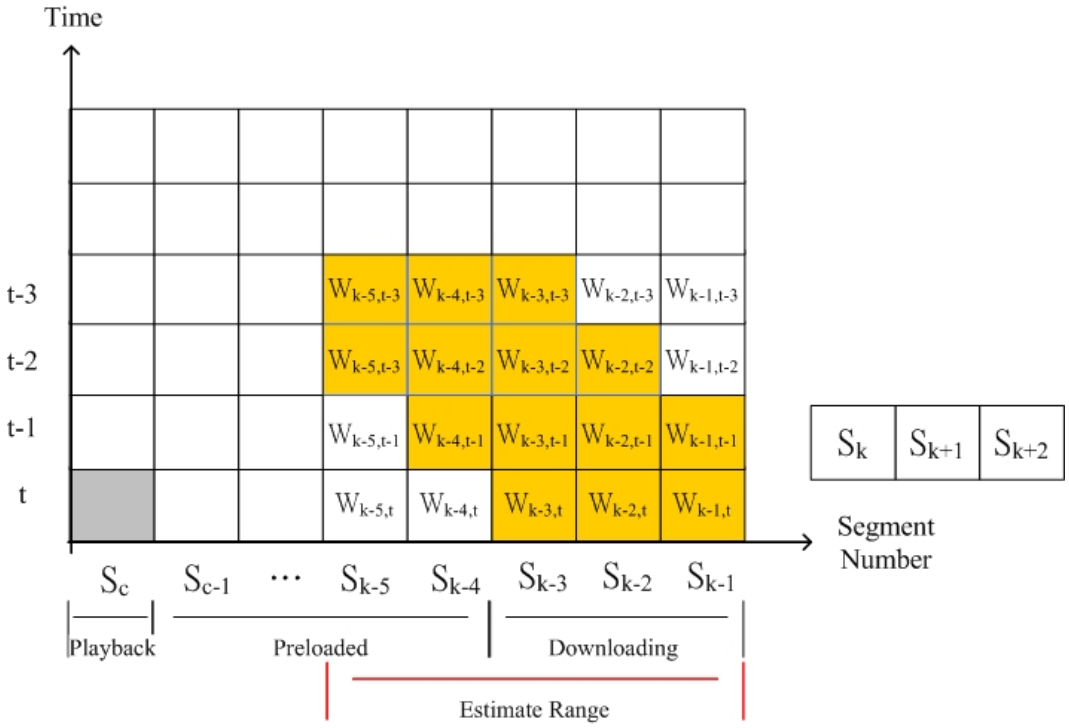


圖 6：預測下載速度之速度分佈

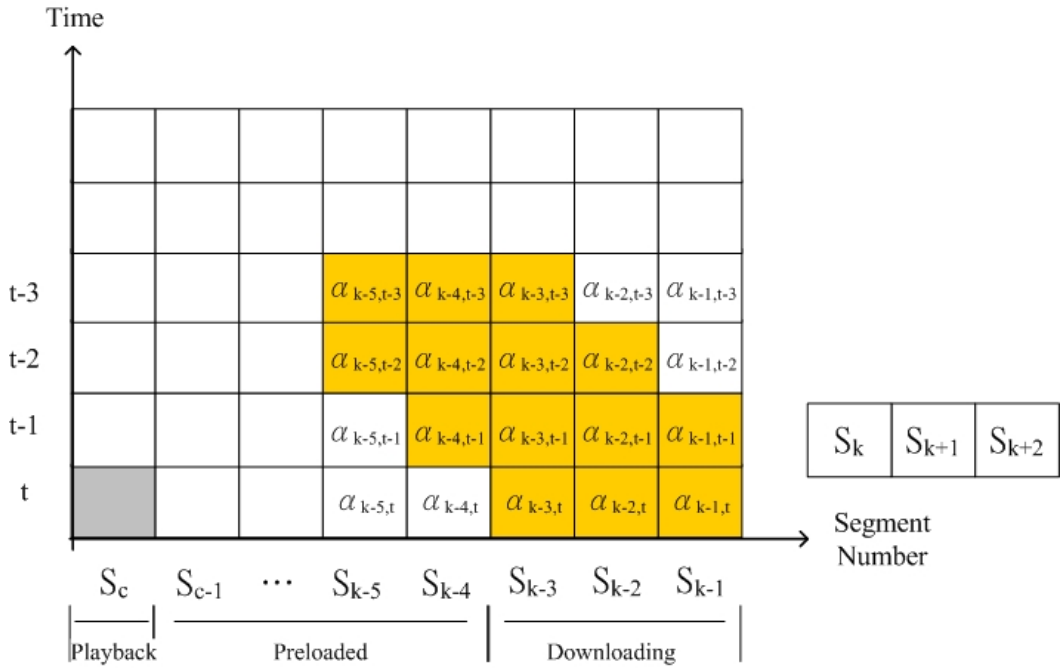


圖 7：預測下載速度之權重分佈圖

圖 7 則是相對之權重分佈情形，故我們可將上述所得出之下載速度計算公式列出：

$$W_k = \sum_{i=1,j=0}^{i\le R,j<Q} \frac{\alpha_{k-i,t-j} W_{k-i,t-j}}{\alpha_{k-i,t-j}}, R = 5, Q = 4$$

接下來探討權重的安排，由於 x 軸的權重被鄰居的差異度所影響，區塊的起始下載時間不同，其鄰居差異度亦有所影響，得到的速度可參考性則隨著起始下載時間的漸遠而大幅降低，故我們在權重方面會以等比級數遞減。

至於 y 軸權重是受到時間的影響，每段時間所得到的速度並不會有太大的變化；唯一的例外即當區塊一開始進行下載時，總是稍微浪費了一些時間在初始化上（與 Tracker 溝通、取得鄰居清單並要求檔案片段等等），故參考性最低。因此，我們以時間 t 得到的權重為基準，每項往前追溯的權重皆以等差級數遞減，而最早的數據我們給予最低的權重。

圖 8 即為我們給予這權重的範例，可發現最底層的目前下載速度是等比級數的比例遞減；再來每往前一段時間的資料，我們給予目前權重再減 1 作為該權重（等差級數）。但當這個資料為最舊時，我們認為此為起始下載的下載資料，故採計可信度最低，給予

的權重也改成只有 1。而預載區間在下載完成後之速度資料為 0，不採計。當我們建構出這個權重表格時，即可一一配合對應的速度值，予以計算出預期下載速度 W_k。

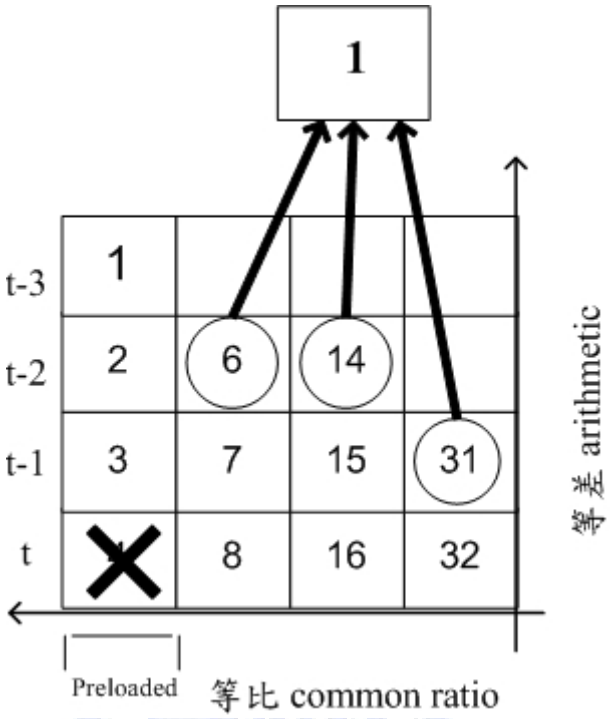


圖 8：預測下載速度之權重範例圖

求出預期下載速度後，為了區塊下載控制的檢查機制，我們便可為這個機制推導出一些式子：

T：區塊的播放時間

N：一份影片被切割的區塊總個數

c：目前播放或是先前播放中斷的區塊編號

m：下載中的區塊個數

k：第 k 個區塊（$1 \le k \le N$）

D_k：第 k 個區塊距離下載完成的時間

F_k：第 k 個區塊尚未下載的檔案大小

W_k：第 k 個區塊的預期下載速度

u：緊急程度，可與緊急指標比較作為是否可開始下載或播放的依據

H.U.：高水線緊急指標（High Urgent Index），可被比較來評斷是否開始下載區塊

L.U.：低水線緊急指標（Low Urgent Index），可被比較來評斷是否開始下載或播放區塊

首先，若要於播放前完成下載動作，必須達成平均每個區塊距離下載完成的時間小於它距離預期播放的時間的假設，其中平均每個區塊的預期下載時間等於該區塊未下載的檔案大小除以其預期下載速度：

$$D_k < (k - c - 1) \times T \tag{1}$$

$$D_k = F_k \div W_k \tag{2}$$

式子(1)右邊是 S_k 距離預期播放還有多久時間，在正常情況下，我們須先計算 S_k 離目前正在播放的區塊還有幾個，便得出$(k - c - 1)$。故式子(2)經過代入式子(1)即成為式子(3)：

$$F_k < W_k \times (k - c - 1) \times T \tag{3}$$

整理後：

$$\frac{F_k}{W_k(k-c-1)T} < 1 \tag{4}$$

由式子(3)可得知該區塊至開始播放前所下載的資料量，必須大於該檔案大小，才能保證順利播放而不中斷。為更加確保中斷情形不會發生，我們使用 H.U.作為一緊急指標並讓 H.U.小於 1，使之更嚴格地對式子(4)的不等式左側把關，以評估是否開始下載，而 H.U.越大表示此標準越寬鬆，H.U.越小則是越嚴格。我們也將不等式左側的比值以 u 表示：

$$u = \frac{\text{距離下載完成的時間}}{\text{距離預期播放時間}} = \frac{F_k}{W_k(k-c-1)T} \tag{5}$$

另外，由於客戶端的下載頻寬由各個下載任務所共享，故考慮到開始新增下載區塊時，會使其它正在下載的區塊的下載速度變慢；若下載中的區塊個數為 m 個，新區塊加入之後為（m+1）個，則預期下載速度應修正為 $\frac{m}{m+1}$ 倍，我們定義：

$$\widetilde{W}_k \stackrel{\text{def}}{=} \frac{m}{m+1} W_k \tag{6}$$

故公式(5)應修正為：

$$u = \frac{F_k}{\widetilde{W}_k(k-c-1)T} \tag{7}$$

若 u>H.U.，表示緊急程度非常高，必須趕緊開始下載；而當 u=H.U.，則是緊急程度中等，再不下載會使 u 提高而使中斷機率上升，故仍開始下載；最後則是 u<H.U. 的情形，我們認為仍在安全範圍內，可暫不開始下載。此外，前文曾提及每次偵測時會從 B 個區塊中挑選，最多只能挑選一個開始下載。若有多個區塊求出的比值 u 皆超出 H.U.時，則選 u 最大的區塊準備開始下載。

當我們選好欲開始下載的區塊（S_k）時，由於會使其它正在下載的區塊下載速度變慢，故必須再度評估：該區塊新增下載動作後，會不會使這些正在下載的區塊因為下載變慢而來不及在播放前完成，造成播放中斷。為考慮此一情形，我們認為必須也求出這些區塊的緊急程度 u，其中下載中區塊的下載速度被新增區塊所分走，平均下降速度為 $\widetilde{W}_k$ 的 m 分之一，因此我們以正在下載的區塊（S_{k-i}）為例，求出新的 $\widetilde{W}_{k-i}$：

$$\widetilde{W}_{k-i} \stackrel{\text{def}}{=} W_{k-i} - \frac{\widetilde{W}_k}{m} \tag{8}$$

再將$\widetilde{W_{k-1}}$代入公式(5)即可得出新的緊急程度 u：

$$u = \frac{\text{距離下載完成的時間}}{\text{距離預期播放時間}} = \frac{F_{k-i}}{\widetilde{W}_{k-i}[(k-i)-c-1]T} \tag{9}$$

得出以上式子後，我們制定幫助判斷 S_k 是否可開始下載的規則：

1. 首先下載中區塊的 u 只要全部都小於 H.U.，S_k 即可開始下載，因為就算 S_k 開始下載，S_{k-i}也不會因為下載速度變慢而造成播放中斷。
2. 但萬一下載中區塊的 u 任一介於 H.U.與 1 之間時（$H.U. \le u \le 1$），則 S_k 不可開始下載，除非 S_k 的 u 大於 1。
3. 任一下載中區塊的 $u > 1$，表示很有可能發生中斷，故 S_k 不能開始下載。

如圖 9 我們整理一些狀況範例來判定 S_k 是否可開始下載動作：

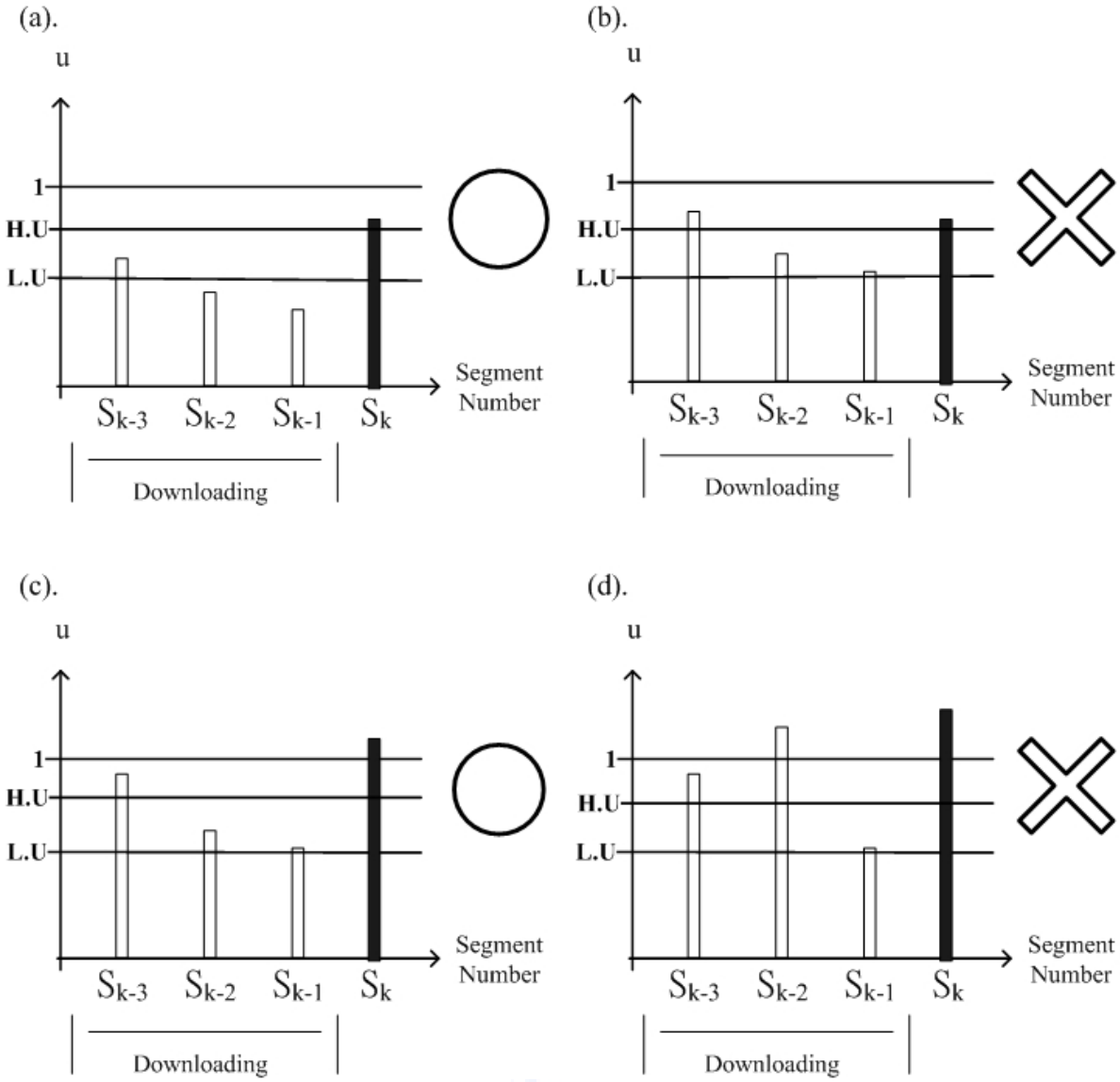


圖 9：判定是否下載之狀況範例圖

圖 9（a）下載中區塊的 u 只要全部都小於 H.U.，表示加入 S_k 也不會使得其它正在下載的區塊下載不及，以至論到它播放時發生中斷。圖 9（b）則是 S_{k-3} 的 u 介於 H.U.與 1 之間，且 S_k 的 u 也不夠高（不夠緊急），故不進行下載動作。圖 10（c）跟圖 9（b）差別只在 S_k 的 u 大於 1，故開始下載。圖 9（d）則是出現下載中區塊的 u 大於 1，此時不可開始下載。

最後我們討論特殊情形：初始化的運作。由於一開始沒有下載速度可參考，無法決定哪個區塊先下載，所以當客戶端一開始剛入網路時先下載第一個區塊。接下來每隔 $\frac{T}{P-1}$ 秒仍檢查是否有區塊須開始下載，但由於目前無區塊播放（$c = 0$），此時：

$$u = \frac{\text{距離下載完成的時間}}{\text{距離預期播放時間}} = \frac{F_k}{\widetilde{W}_k(k-c-1)T} = \frac{F_k}{\widetilde{W}_k(k-1)T}$$

由於挑出預下載區塊後，仍須評估新區塊加入後是否仍穩定，故對正在下載的區塊算 u；其中因為第一個區塊的距離預期播放時間為零無法計算 u，故計算是否下載時，我們將其忽略。

至於初始化的播放時機，由於直到第一個區塊下載完成才檢查是否開始播放，此時依然如偵測下載般算出下載中與欲下載的區塊的 u。接著為避免未來發生播放中斷情形，我們制定一低水線的標準（L.U.），所有計算出的 u 都必須小於 L.U.才可開始播放。若檢查後仍未播放，則在每次檢測是否下載的同時也一起檢查是否播放，直至第一個區塊開始播放後，系統恢復正常如前。

第二種特殊情形為下載狀況非常好時，下載區間往往沒有區塊在下載；為充份利用下載頻寬，我們將算出下載中區塊的平均 u，若平均 u 小於 L.U.且其中最大 u 亦小於 H.U.時，則一定下載區塊，此時再另外從欲下載區塊中挑出 u 最大的開始下載（不需與 H.U.比較）。

最後我們列出完整之流程圖（不包含初始化狀況），如圖 1

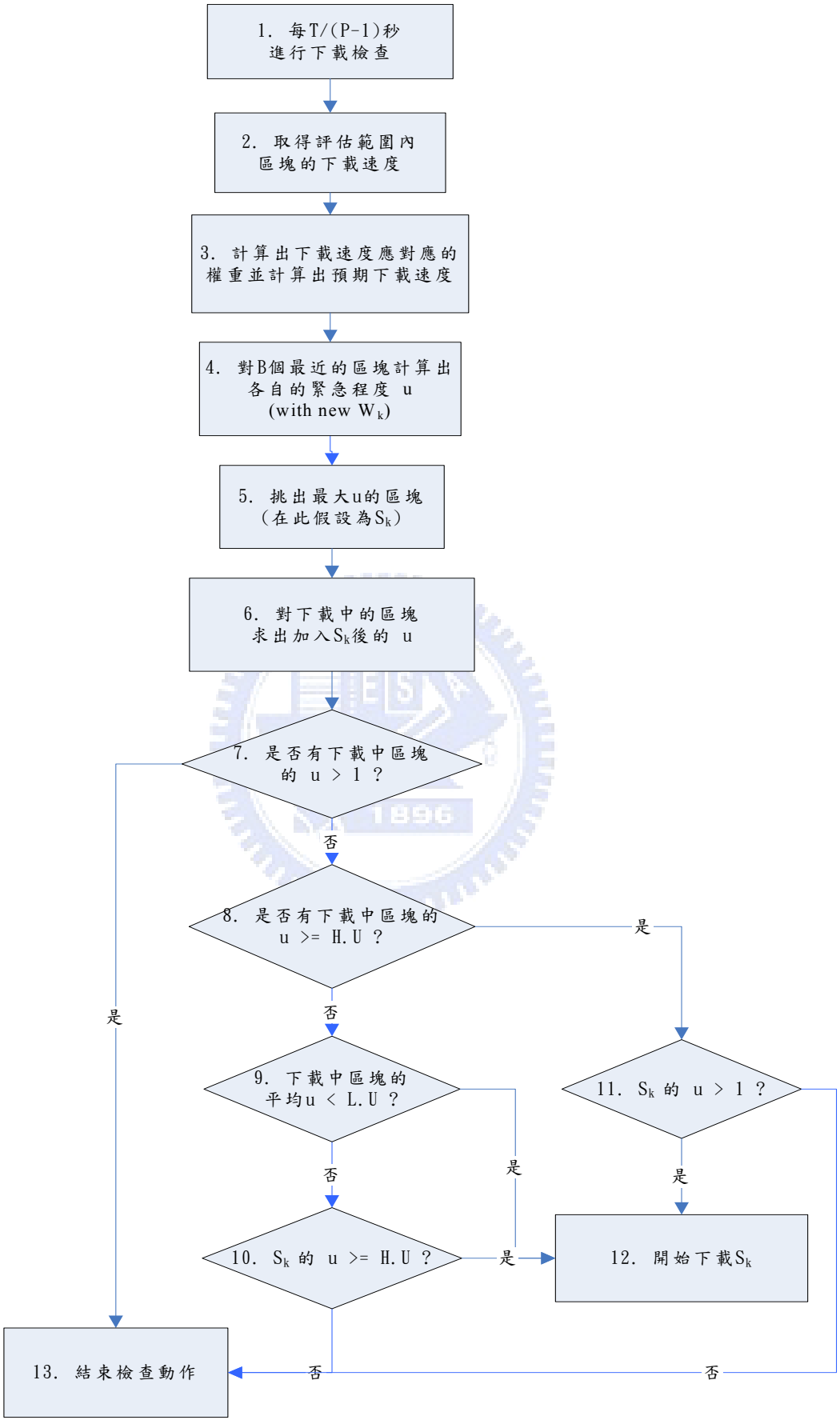

1. 每T/(P-1)秒
進行下載檢查
2. 取得評估範圍內
區塊的下載速度
3. 計算出下載速度應對應的
權重並計算出預期下載速度
4. 對B個最近的區塊計算出
各自的緊急程度 u
(with new W_k)
5. 挑出最大u的區塊
(在此假設為S_k)
6. 對下載中的區塊
求出加入S_k後的 u
7. 是否有下載中區塊
的 u > 1 ?
否
8. 是否有下載中區塊的
u >= H.U ?
是
否
9. 下載中區塊的
平均u < L.U ?
是
否
10. S_k 的 u >= H.U ?
是
11. S_k 的 u > 1 ?
是
12. 開始下載S_k
是
否
否
13. 結束檢查動作

圖 10：偵測下載流程圖

3.4 作種區塊排程演算法

在本節中提出我們的作種排程演算法，作種演算法的設計目的在於控制客戶端的作種區塊分配來提升整體同儕網路的作種能力，並希望能協助伺服器端，甚至達到多個伺服器的分享能力。

由於客戶端的播放有連續性（依順序播放），意謂每個客戶端在預載區間至播放區間所擁有的區塊介於一個範圍內，隨著時間流動，這些連續播放的區塊被趨近於同一批的鄰居同儕所維護與服務。如圖 11 所示，不同的時間，即使下載中的區塊隨著時間變化而改變，只要每個同儕的下載時間接近，提供的同儕或鄰居亦不會有太大變化。

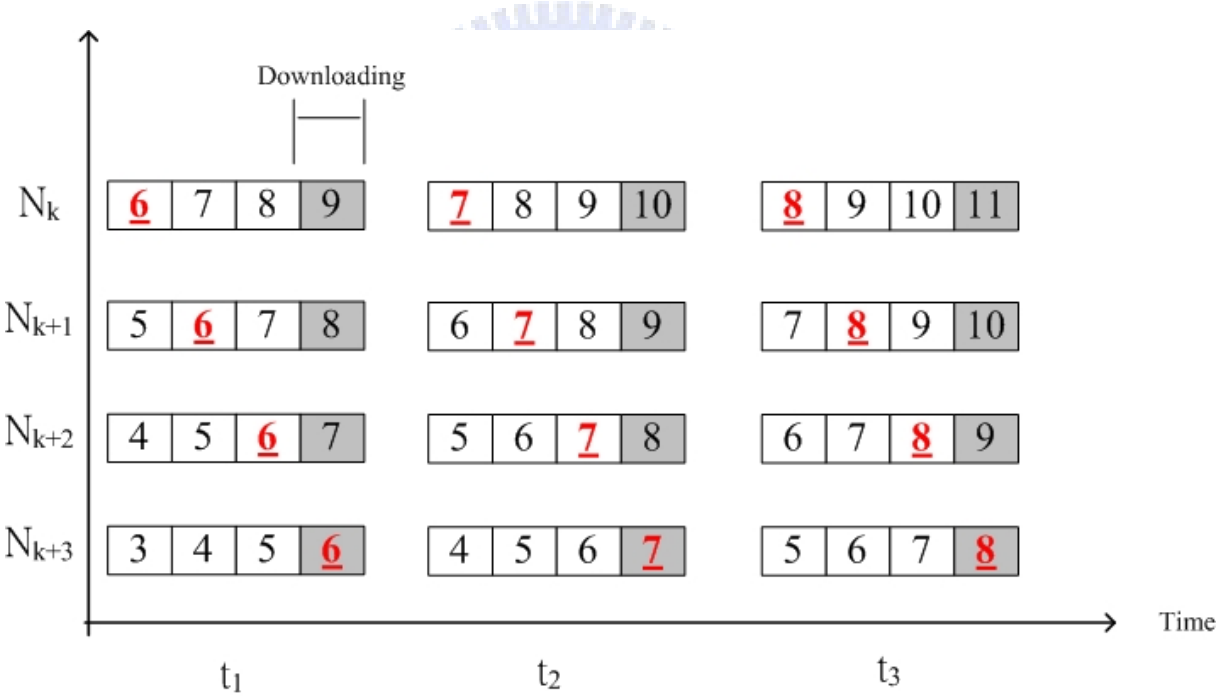


圖 11：播放連續性造成之鄰居相近之示意圖

整體而言，隨著不同時間進入網路的同儕，其分享作種的區塊分佈可推測為一種均勻的分佈交疊狀態。由於作種分佈的連續性，同儕的到達情形深深地影響著整體作種分佈是否均勻。若欲增加單一伺服器之分享能力，提高可服務之客戶數，就必須使作種分享的區塊均勻地分散交疊，成為多個伺服器的分享能力，也就是使其分享能力提升了數倍。

由於客戶端上傳頻寬與緩衝區空間有限，當作種區間的個數超出限制時，我們則挑出最不重要的區塊進行剃除，不再作種，而[13]認為快取空間的替代策略越簡單越佳。因此本論文在此將作種排程演算法使用 LRU（Least Recently Used）以符合此需求；也就是作種區間即將溢位時，用 LRU 計算出最不常上傳分享的區塊並移除。此作法可使作種的區塊分佈較為分散，不至於過度集中於某幾個連續區塊。

Chapter 4 實驗分析

在此章節將分析本演算法之表現結果，4.1 節說明模擬設定、背景與欲觀察之資料，4.2、4.3 節將就圖表分別探討本演算法所呈現之優劣。

4.1 實驗環境設定

本論文在此使用PeerSim[14]作為模擬的平台，這是一個相當適合模擬P2P行為的模擬器；程式碼以Java撰寫，故能運作在不同的作業平台。且PeerSim易於擴充，程式碼撰寫十分模組化，適合用於各種大型專案之開發。

PeerSim 使用兩種不同的模擬模型，分別是 Cycle-based 與 Event-based。Cycle-based 適合大規模的模擬，佔用資源較少，但忽略傳輸層堆疊上的模擬，故只適合於較簡易之模擬應用。而 Event-based 引擎的可操控靈活度與真實度較高，可動態地控制每個同儕的運作，雖然可操控規模較低，但適合複雜性高的模擬應用。

本論文在此使用Event-based模型作為主要開發的架構，使用版本為 1.0.4。並選用實作在PeerSim上的BitTorrent模組專案[15]，在此基礎上進行修改，延伸開發本論文所發表之演算法。

在此將對整個實驗模擬給予適當的環境，再針對演算法之實作方面作模擬的參數設計與效能評估。首先在實驗環境部份，我們先設定一些基本環境參數如：

- 影片總長度：40 分鐘
- 總模擬時間：120 分鐘
- 每個區塊的播放時間（T）：固定 20 秒
- 一份影片被切割的區塊個數（N）：120 個
- 每個區塊的播放位元率（Bit Rate）：400～700Kbps
- 每個區塊的檔案大小（F_k）：0.8～1.5MB
- VoD 伺服器端在實驗中只放置一個

- 下載方式為 BitTorrent 網路協定，基礎設定亦同
- VoD 伺服器端的網路頻寬設定為 50Mbps/50Mbps
- VoD 客戶端實際可使用頻寬為最大網路頻寬的 80%，頻寬分佈如表格 1：

表格 1：實驗網路設定表

	最大網路頻寬		實際可使用頻寬
頻寬分佈	高（50Mbps／50Mbps）：10% 中（10Mbps／2Mbps）：50% 低（3Mbps／1Mbps）：40%	× 80%	**40Mbps／40Mbps** **8Mbps／1.6Mbps** **2.4Mbps／820Kbps**

4.2 下載區塊排程演算法之實驗設計與分析

我們以使用者的觀看狀況作為衡量的指標，並求出最佳之參數設置。最主要的兩項衡量指標為：

$$\text{Avg}\left(\frac{\text{播放中斷次數}}{\text{區塊個數}}\right)、\text{Avg}\left(\frac{\text{播放中斷時間}}{\text{區塊個數}}\right)$$

其它實驗設定：

- 偵測是否下載的時間週期（$\frac{T}{P-1}$）：$\frac{T}{4}$
- 取得區塊下載速度時，區塊的編號範圍（R）：5 個
- 取得區塊下載速度時，往前追溯之時間次數範圍（Q）：4 個
- 客戶端進入網路的到達率（Arrival Rate）：穩定且高：約每 6 秒有一人進入網路（抵達間隔時間約 6 秒），當有客戶端節點播放完成後，客戶端的到達率與離開率約莫相等，此時網路中客戶端的總數趨於穩定約為 500
- 作種區間之區塊個數設定，以占總檔案大小（總區塊個數）的比例（x％）為參數。為不影響下載效率之判斷，在此設定客戶端的作種區間大小比例：100%

接下來是我們的實驗變數：

◆ 檢查是否開始下載的區塊數（B）：1～3 個

◆ 判斷新增下載及播放的時機時，會以計算出的緊急程度（u）與緊急指標比較；因此緊急指標的高低水線（H.U.、L.U.）成為判斷區塊是否應該下載的重要依據：

1. H.U.＝0.8、L.U.＝0.5（差距為 0.3）
2. H.U.＝0.8、L.U.＝0.6（差距為 0.2）
3. H.U.＝0.8、L.U.＝0.7（差距為 0.1）
4. H.U.＝0.9、L.U.＝0.6（差距為 0.3）
5. H.U.＝0.9、L.U.＝0.7（差距為 0.2）
6. H.U.＝0.9、L.U.＝0.8（差距為 0.1）

我們由檢查下載的區塊數（B）與高低水線（H.U.、L.U.）的調整來觀察平均播放中斷次數與時間的變化。由圖 12、13 可發現高低水線固定，B 為 2 時，表現也較其他兩者（B＝1、B＝3）稍佳。而偵測下載區塊數固定，高低水線的差距越大時表現也較佳。當偵測下載區塊數與高低水線的差距固定，高低水線相對較低的整體播放中斷次數與時間皆下降，下載表現較佳。

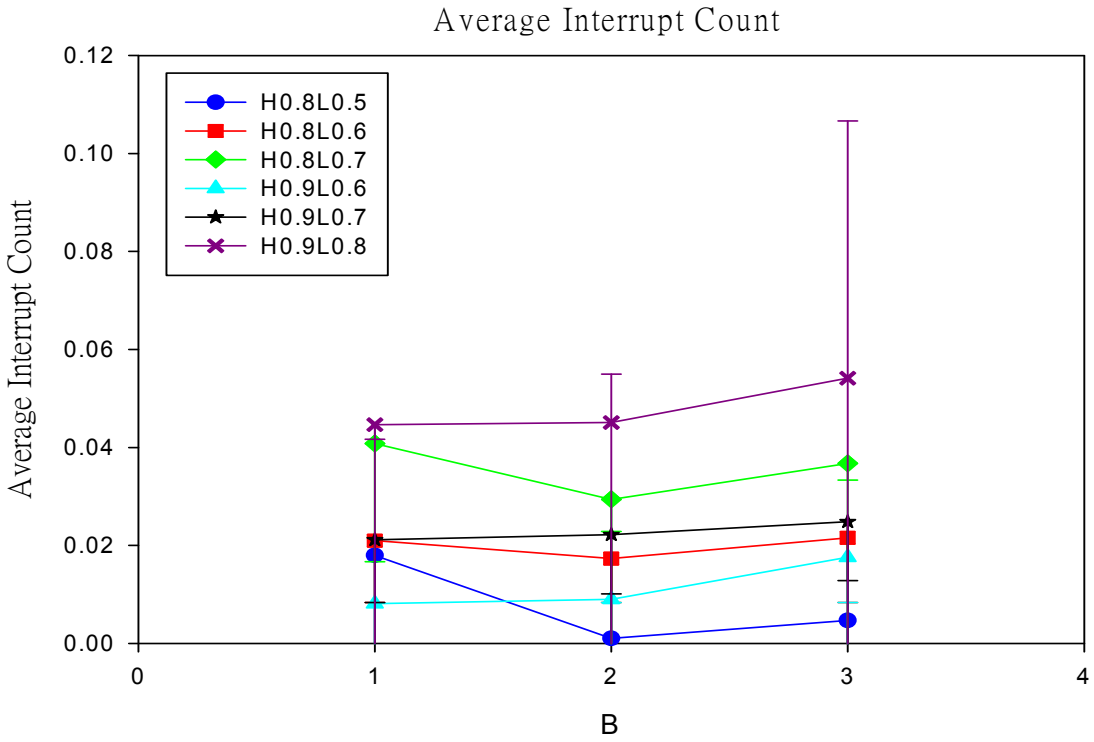


圖 12：平均中斷次數比較圖

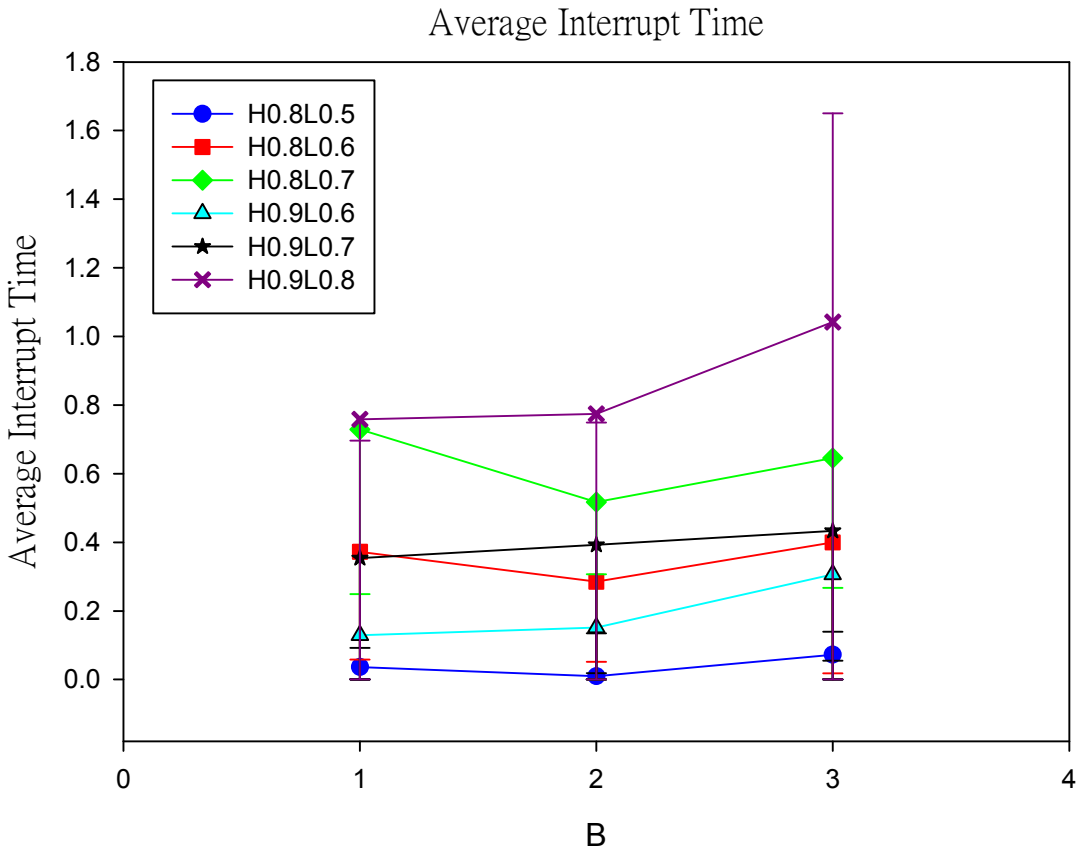


圖 13：平均中斷時間比較圖

圖 14～17 為針對中斷情形最好（B＝2、高水線 0.8、低水線 0.5）與最差（B＝1、高水線 0.8、低水線 0.7）作總中斷時間的個數分組交叉比對。雖然絕大多數的中斷時間皆在 10 秒以下，但可發現表現差的設定有較多中斷情形異常高的比例。且高低水線的設定影響比一次偵測多少個區塊下載之影響來得大。

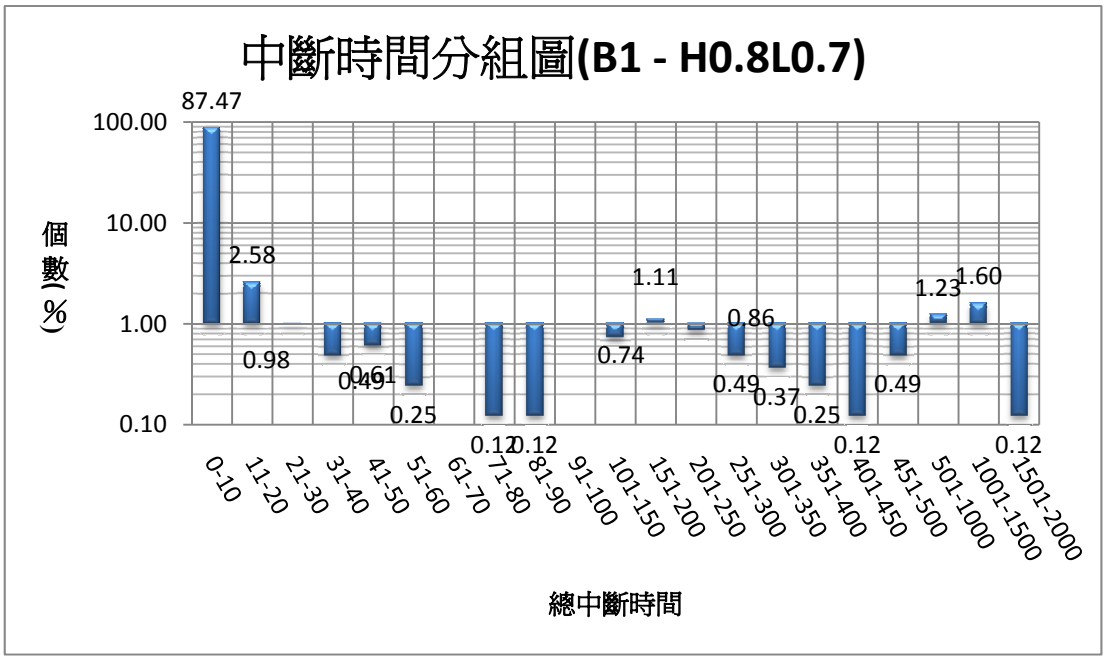


圖 14：中斷時間分組圖（B1 - H0.8L0.7）

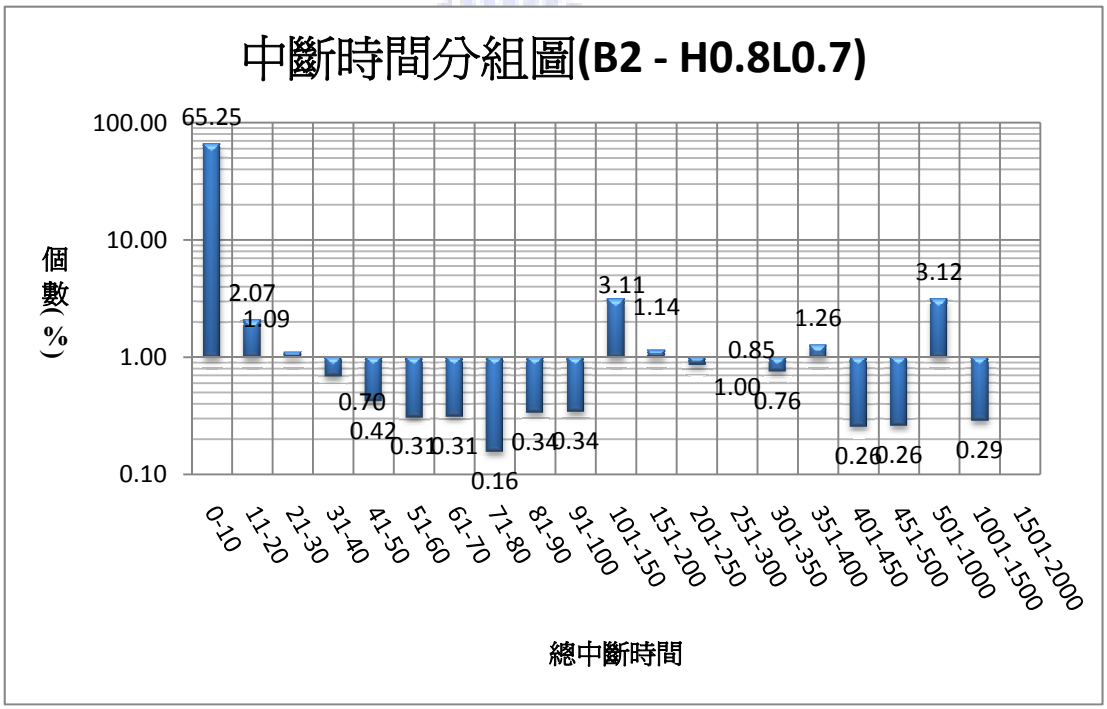


圖 15：中斷時間分組圖（B2 - H0.8L0.7）

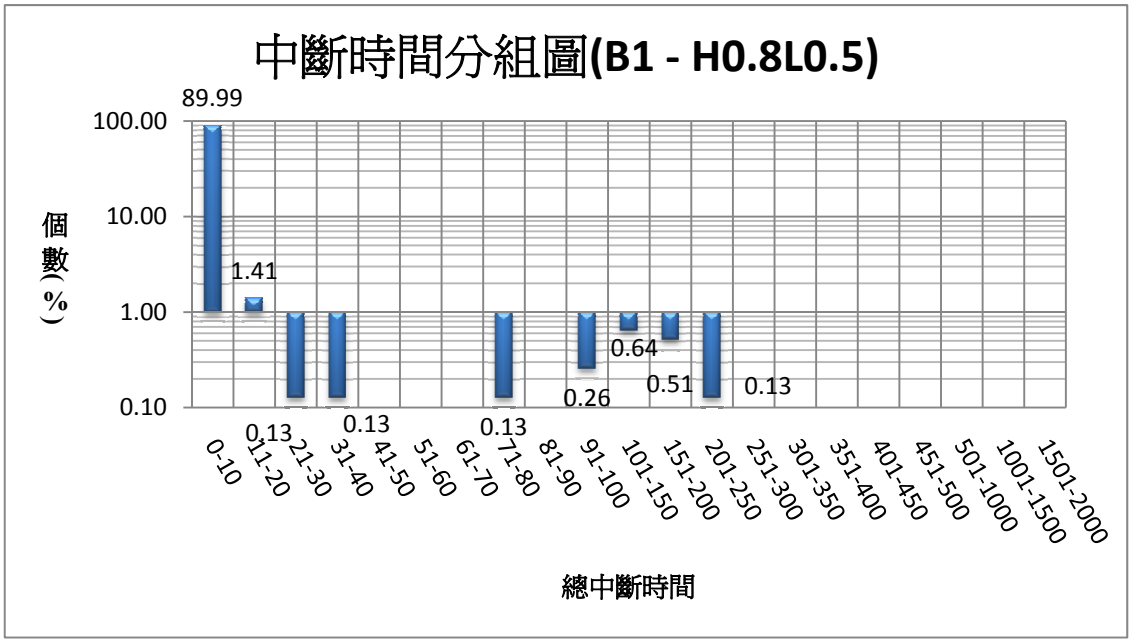


圖 16：中斷時間分組圖（B1 - H0.8L0.5）

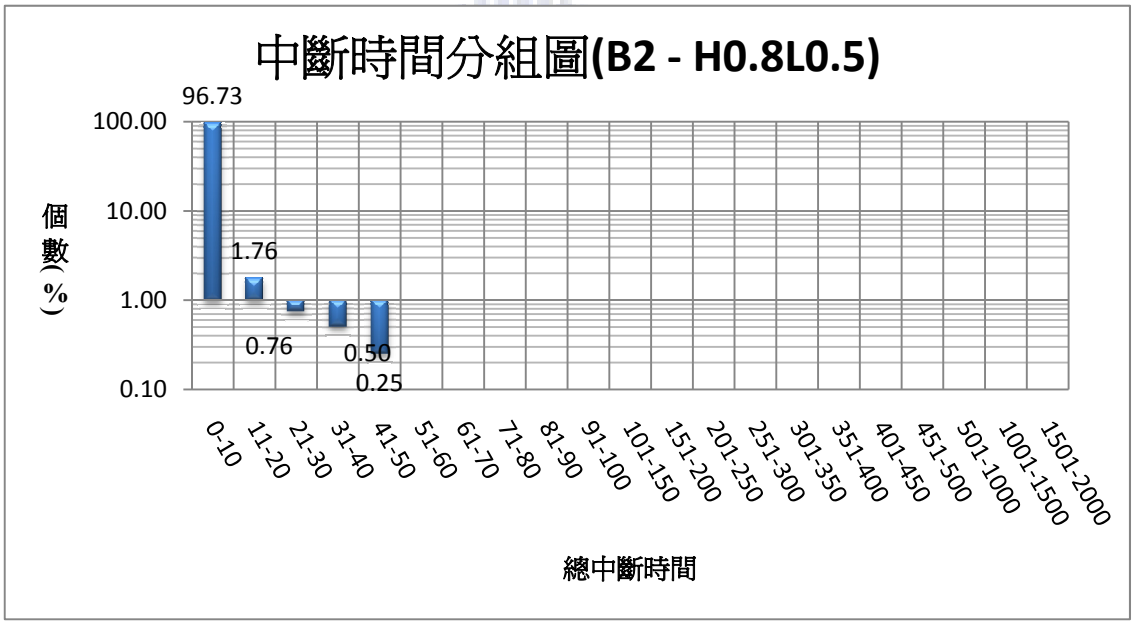


圖 17：中斷時間分組圖（B2 - H0.8L0.5）

至於圖 18 則是以啟始播放延遲（Start-up Delay）作為比較，從中可發現當低水線越高時，判斷是否播放的標準也越寬鬆，故較早開始播放，啟始播放時間也較低。

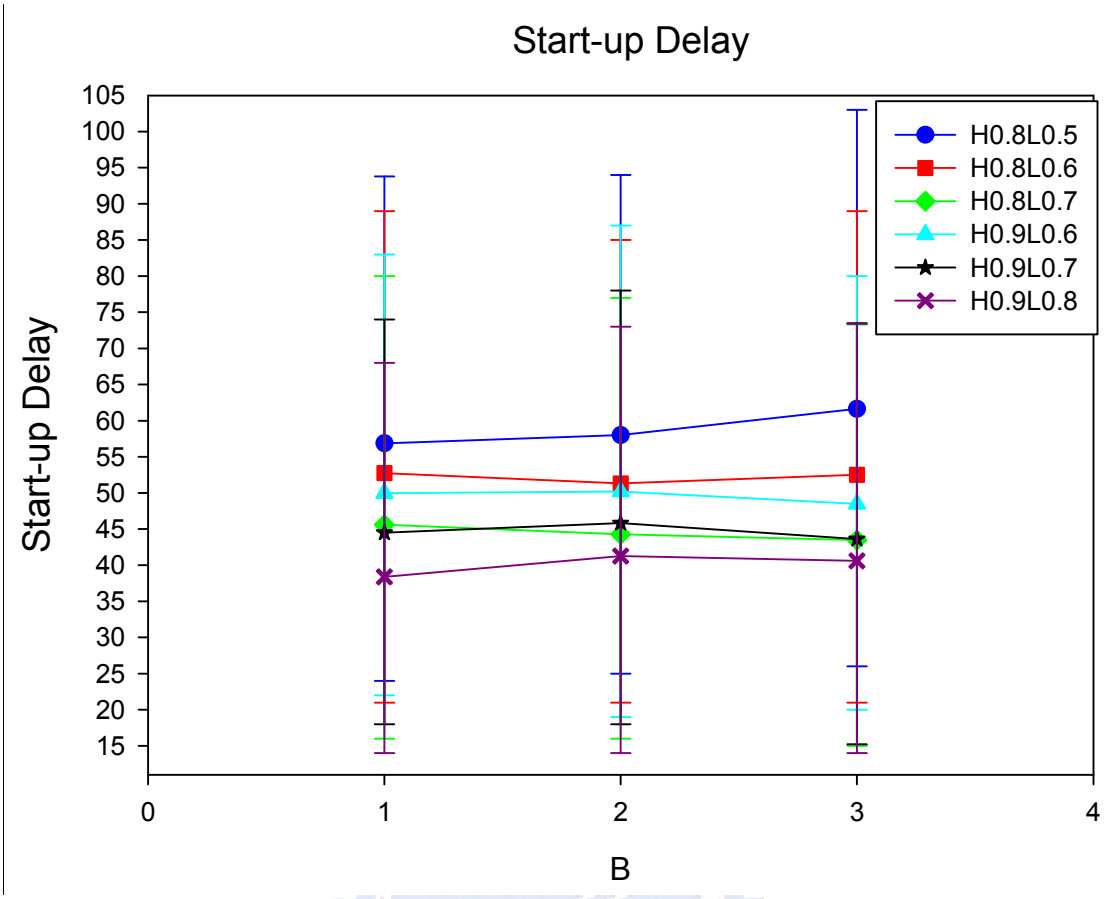


圖 18：平均啟始播放延遲時間比較圖

由以上三項觀察，我們將 B 設定為 2，L.U.設為 0.5 為最佳之參數設定，並藉此探討作種演算法之影響。

4.3 作種區塊排程演算法之實驗設計與分析

在此我們想觀察作種演算法在不同作種區塊區間個數時，是否能維持使用者的觀看狀況，進而達到維持既有下載效率，並能以最佳方式分配作種的區塊。

在本實驗的基礎設定為：

- 偵測是否下載的時間週期（$\frac{T}{P-1}$）：$\frac{T}{4}$
- 取得區塊下載速度時，Segment 的編號範圍（R）：5 個
- 取得區塊下載速度時，往前追朔之時間次數範圍（Q）：4 個
- 檢查是否開始下載的區塊數（B）：2

- 緊急指標的高低水線（H.U.、L.U.）：H.U.＝0.8、L.U.＝0.5

最後是實驗變數，本論文將交叉觀察不同同儕到達率與作種區間大小比例之使用者觀看效果：

◆ 同儕到達率：

1. 低落：到達率較低，抵達間隔時間約每 30 秒有一人進入網路
2. 穩定且高：抵達間隔時間約每 6 秒有一人進入網路
3. 熱門：初期低落，約 60 分鐘後觀看人數暴漲，再過 30 分鐘後人數趨減

◆ 作種區間個數的設定，我們以占總檔案大小（總區塊個數）的比例（x％）為參數，表示每$(100 \div x)$個客戶端節點的區塊分佈交疊起來應能成為一個伺服器的分享能力。我們在此設定客戶端的作種區間大小比例為：5%～50%

◆ 另一種設定方式為依頻寬設定：

1. 高：80%、中：20%、低：5%（頻寬越高，作種比例越高）
2. 高：5%、中：20%、低：80%（頻寬越高，作種比例越低）

我們以圖 19、20 觀察在不同到達率與不同作種比例來觀察平均播放中斷次數與時間的變化。由於同儕到達率為熱門的狀況較為特別，當某個特定時段的節目特別多人觀看，會使大量的人進入隨選視訊系統，這種瞬間大量湧入的狀況會使整體網路的分享狀態不足以負荷下載需求，也使得中斷次數與時間比穩定與低落的同儕到達率高。至於不同作種比例方面，隨著作種比例的增加，同儕之間可分享的區塊也增加，使得作種的區塊分佈可成為一個甚至多個伺服器的分享能力，分享能力提高後，客戶端下載效率也較好，因此可發現平均播放中斷次數與時間也隨之降低；但隨著作種比例提高，中斷次數與時間的降幅也漸漸趨緩。

另外，圖中也顯示依頻寬安排作種比例的比較。由於三種頻寬站網路中比例不同，故高：80%、中：20%、低：5%的實際作種數恰為整體網路皆固定作種比例 20%之配置（$10\% \times 80\% + 50\% \times 20\% + 40\% \times 5\%$）；而高：5%、中：20%、低：80%為固定 42.5%。

為較易比對，本論文將這兩種依頻寬調整的方式整理成固定比例之方式。首先可發現頻寬越高且作種比例越高的安排方式，比頻寬越高但作種比例越低還要好。另一方面頻寬越高且作種比例越高亦可與整體作種比例 20%的中斷情形比較，可發現雖然在網路中的總作種比例相同，但頻寬越高且作種比例越高的中斷情形卻比固定 20%好上許多，這是因為頻寬越高且作種比例越高的方式更有效利用高頻寬的優勢而不浪費。反之頻寬越高且作種比例越低的中斷情形與整體 40%比較，中斷情形嚴重許多，表示高頻寬的高上傳速度被浪費掉。這些在整體網路需求越高時（熱門與穩定且高之到達率）差距越明顯，反之則無異。

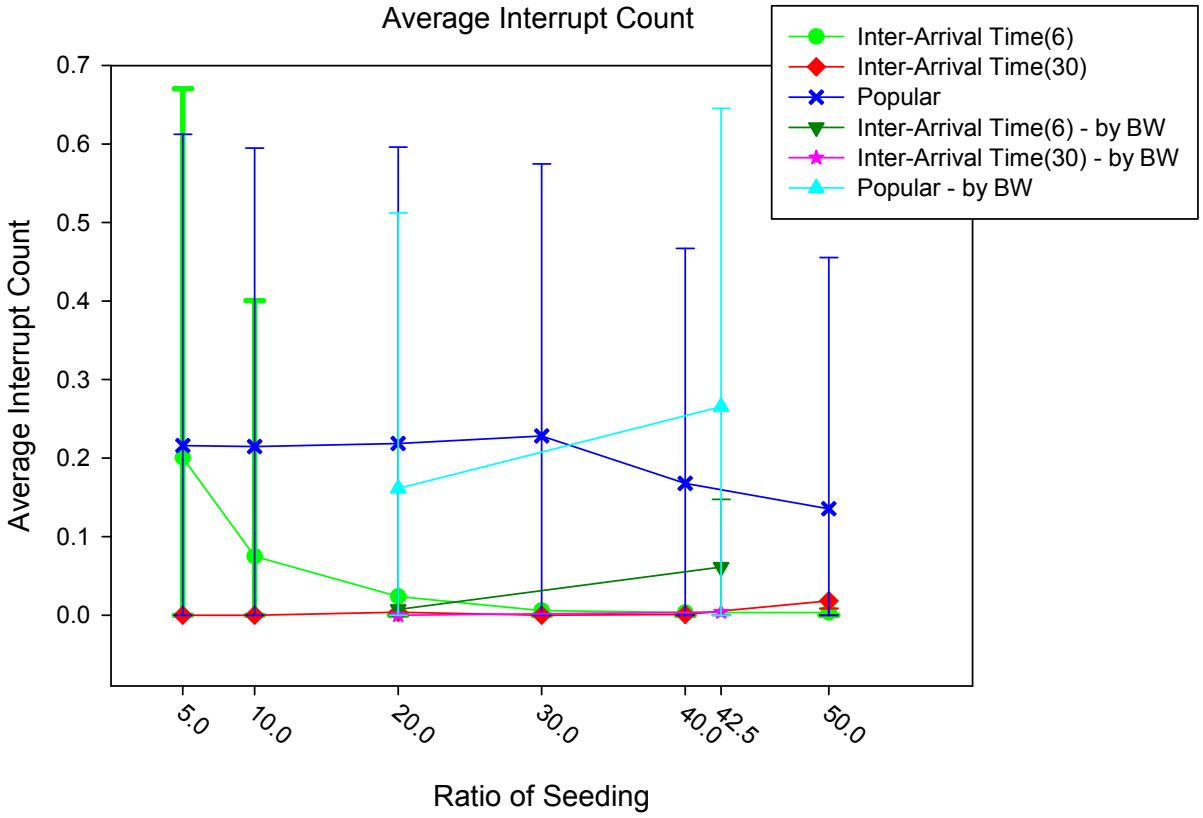


圖 19：不同作種比例之平均中斷次數比較圖

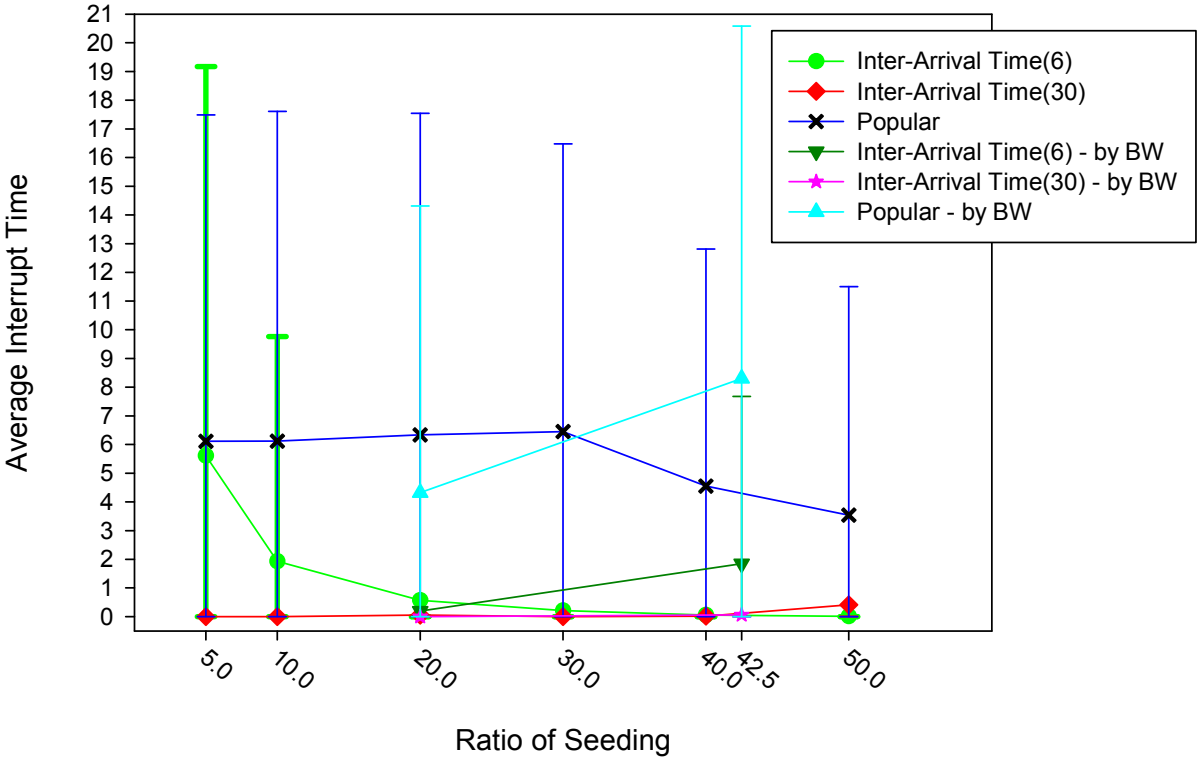


圖 20：不同作種比例之平均中斷時間比較圖

圖 21 顯示啟始播放延遲之比較，可以發現抵達間隔時間為 30 秒的曲線隨著作種比例上升而增加，但其他卻是漸漸下降甚至持平。這是因為間隔時間 30 秒的人數少，可由伺服器支援，故啟始播放延遲可較低，但其他兩種到達率則因為線上客戶數過多，伺服器提供不足且作種比例低時無法完整支援，因此啟始播放延遲較高；而當作種比例上升時，到達率低落的選擇較多，有可能向上傳頻寬低的同儕要求檔案，故啟始播放延遲提升。

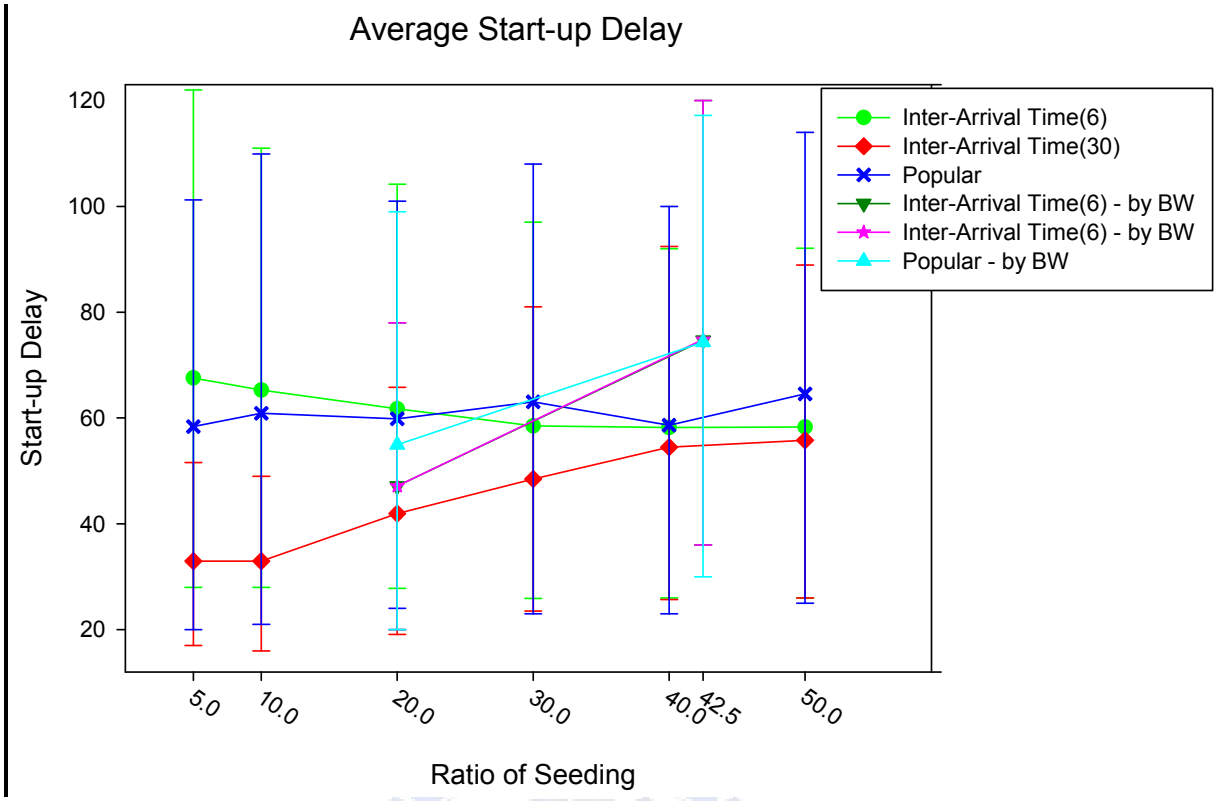


圖 21：不同作種比例之平均啟始播放延遲時間比較圖

在此我們將客戶端實際上傳量除以伺服器上傳分享量，可粗略得出這個作種比例會產生多少伺服器的分享能力（含伺服器本身），圖 18 即為在不同作種比例之分享能力比較圖。就三種到達率觀察，可以發現穩定且高的到達率之分享能力最高，熱門的情況次之，最後則是最差的低落狀況，足見同時線上同儕數對於整體網路的分享能力影響甚鉅。另外我們各種到達率不同作種比例來看，低落情形的分享能力一直趨於穩定，表示因為人數少，只要少許的作種個數即可使分享能力穩定，故作種比例設定為本實驗中最低的 5%即可。而熱門情形則是很緩慢地成長直至作種比例為 40%，我們亦觀察平均中斷次數與時間的變化，也是以高頻寬高作種比例表現較佳，但仍低於穩定到達率之分享能力，故發生中斷之機率較高。最後是穩定的到達率，由於線上同儕數多，客戶端需求亦高，可發現分享能力至作種比例 30%才趨於穩定，同時我們比照平均中斷次數與時間，他的趨勢走向跟本圖皆是從作種比例 30%趨於平穩，故作種比例 30%為本實驗穩定且高到達率之最佳作種比例。最後我們觀察依頻寬分配的分享能力並不十分突出，這是因為低頻寬客戶端的個數較多，總和分享看起來會較差的緣故。

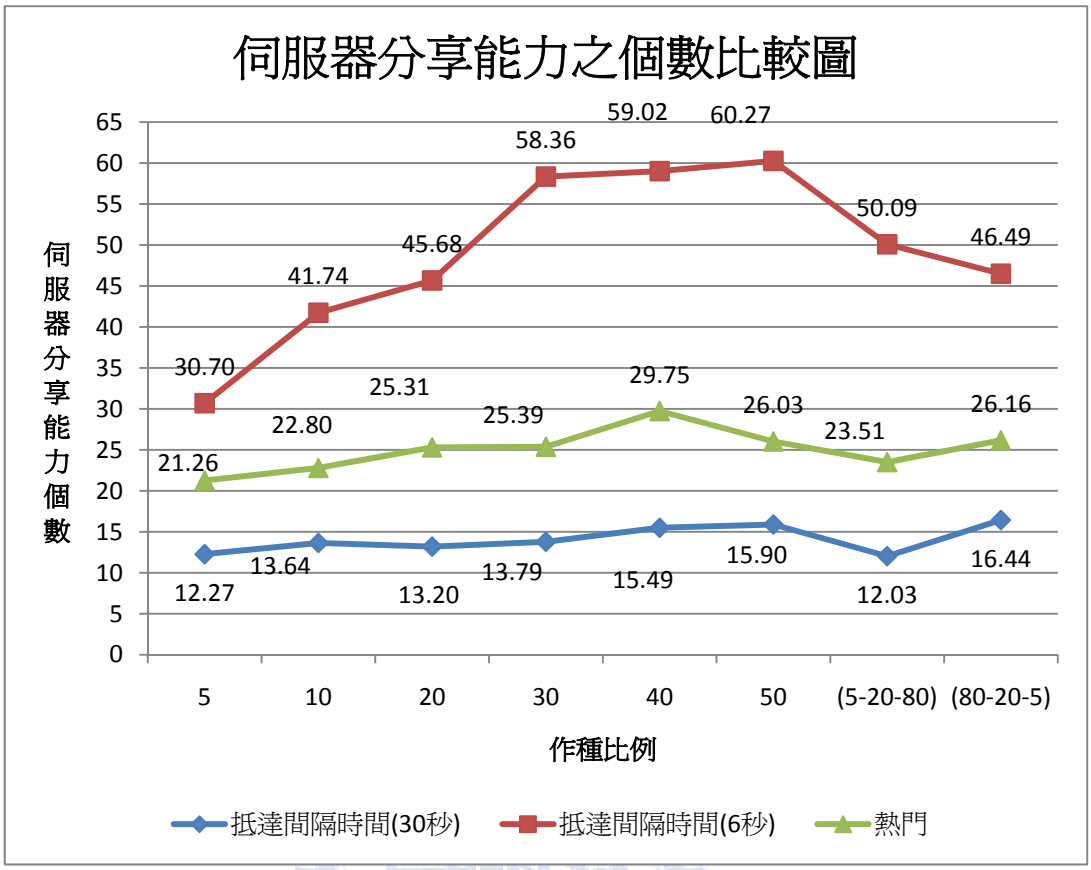


圖 22：不同作種比例之分享能力比較圖

最後本論文比較依頻寬分配作種比例與每個客戶端皆相同作種比例的各種效能。本論文發現在平均中斷次數與時間方面有很顯著的差異，但當作種比例分配為 50%時，依頻寬安排的效果顯得沒有比較好。但我們考慮到依頻寬分配作種比例的啟始播放延遲卻比作種比例分配為 50%少十幾秒。表示在十分接近之觀看品質的立足點上，使用者可更早開始收看影片；且依頻寬分配可更有效分享區塊，讓有能力之同儕付出更多。且加上觀察作種分享能力後，高且穩定的到達率並不需要將作種比例加到 50%，只要 30%即可。因此依頻寬分配作種比例（高 80%－中 20%－低 5%）為本實驗之最佳設定。

Chapter 5 結語

本章節中將對本論文之研究作出結論，並探討未來如何能更增進本研究之可靠度與更周全之分析。

5.1 研究成果

本研究的主要貢獻有下列幾點：

- 提出一區塊作種排程演算法，該演算法利用作種區塊個數的變化而有不同的選擇，並增進隨選視訊系統伺服器之分享效能，可因應大量之使用者需求。
- 客戶端的區塊下載排程演算法，使用加權移動平均法推測未來該區塊可能之下載速度，對下載排程的決定有重大影響。
- 對於客戶端下載個數的控制排程演算法，以緊急指標的高低水線設定不同之影響最大。
- 本論文利用修改之後的 BitTorrent 模擬模組進行模擬，得到結果如下：
  - 下載排程演算法中緊急指標的低水線越低或高低水線差距越大，對整體的播放中斷次數與時間有較佳之影響，下載表現較佳。
  - 作種排程演算法的作種個數安排，以高頻寬配合較多作種個數會表現最佳之結果，最能因應各種同儕到達狀況。

5.2 未來方向

本研究中仍然有一些尚待精進之工作及可行之研究方向，將能使本論文更趨完整，如下所示：

- 對於作種排程演算法的作種個數安排，依頻寬安排之比例可有更多樣的比例調整，藉此找出更佳之參數設定。

- 對於下載排程演篹法之緊急指標高低水線，可有更多樣的比較。
- 對於下載排程之預期速度權重，可更加深入研究，預期下載速度推測得越精準，對於下載演算法之效能更佳。
- 基於本論文之設定下，可再研究如何將啟始播放延遲更為降低。

參考文獻

[1] Video on Demand wiki. http://zh.wikipedia.org/wiki/VOD .

[2] C. Huang, J. Li, and K. W. Ross. “Can Internet Video-on-Demand be Profitable?” In Proceedings of ACM SIGCOMM’07, Kyoto, Japan, August 2007.

[3] http://zh.wikipedia.org/zh-tw/P2P .

[4] The Official BitTorrent Home Page. http://www.bittorrent.com/ .

[5] BitTorrent wiki (protocol). http://en.wikipedia.org/wiki/BitTorrent

[6] BitTorrent Specifications. http://wiki.theory.org/BitTorrentSpecification .

[7] C. Dana, D. Li, D. Harrison, C. N. Chuah. “BASS: BitTorrent Assisted Streaming System for Video-on-Demand,” in IEEE international Workshop on Multimedia Signal Processing (MMSP), IEEE Press, Oct. 2005.

[8] CoolStreaming. http://www.coolstreaming.org/ .

[9] Bo Li; Yang Qu; Yik Keung; Susu Xie; Chuang Lin; Jiangchuan Liu; Xinyan Zhang "Inside the New Coolstreaming: Principles, Measurements and Performance Implications Powerpoint presentation" IEEE Infocom 2008, Phoenix, AZ, A.

[10] A. Vlavianos, M. Iliofotou, and M. Faloutsos, “BiToS: enhancing BitTorrent for supporting streaming applications,” Proc. 9th IEEE Global Internet Symposium, 2006.

[11] http://en.wikipedia.org/wiki/PPLive .

[12] http://en.wikipedia.org/wiki/PPStream .

[13] J. Wu, B. Li, “Keep Cache Replacement Simple in Peer-Assisted VoD Systems” in Proceedings of INFOCOM 2009

[14] http://peersim.sourceforge.net/ .

[15] http://peersim.sourceforge.net/code/bittorrent.tar.gz .