國立交通大學

網路工程研究所

碩 士 論 文

限制延遲時間的點對點即時影音串流系統

A Delay-Constraint P2P Live Streaming System

研 究 生：呂孝恆

指導教授：易志偉 教授

中 華 民 國 一百 年 六 月

限制延遲時間的點對點即時影音串流系統

A Delay-Constraint P2P Live Streaming System

研 究 生：呂孝恆　　Student：Hsiao-Heng Lu

指導教授：易志偉　　Advisor：Chih-Wei Yi

國 立 交 通 大 學

網 路 工 程 研 究 所

碩 士 論 文

A Thesis
Submitted to Institute of Network Engineering
College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
In

Computer Science

June 2011

Hsinchu, Taiwan, Republic of China

中華民國 一百 年 六 月

限制延遲時間的點對點即時串流系統

學生： 呂孝恆　　指導教授： 易志偉

國立交通大學 資訊工程學系 網路工程所 碩士班

摘 要

內容散佈服務(Content distribution services, CDSs)已成為今日網際網路的主要流量來源，而根據統計，其中一半的流量來自影音串流服務。即時影音串流(Live streaming, LS)，提供人們一個平台用來即時分享身邊正在發生的事件，將成為未來主要的影音服務之一。Playback delay，即事件發生後相隔多久時間才能被其他使用 LS 服務的使用者觀看到，是評量一個 LS 系統最重要的依據，特別是當提供 HD 品質等高頻寬需求的 LS 服務時。因此，本篇文章專注於研究如何在樹狀 P2P 網路中提供 LS 服務，使得系統能夠在限制 Playback delay 上限值與提供高串流資料率時服務更多的使用者。我們提出了兩個演算法的組合，PCL 生成演算法及區域最佳化演算法。PCL 生成演算法讓新加入網路的節點能夠被分配到適當的樹狀網路位置，使得能貢獻較多上載頻寬的節點能夠經歷到較小的 Playback delay。而區域最佳化演算法則提供系統一個分散式的方式來優化系統網路，用以降低系統平均 Playback delay。模擬結果顯示，相較於僅考慮 Playback delay 的演算法，我們所提的演算法組合能服務較多的使用者，尤其是提供高串流資料率與限制 Playback delay 上限時表現尤其突出。除了模擬工作以外，我們實作了一個基於樹狀 P2P 網路的 LS 系統雛型。該系統雛型讓我們能以真實網路環境來驗證與改進所提演算法，並已實際被應用在教學用途上。

A Delay-Constraint P2P Live Streaming System

Student: Hsiao-Heng Lu Advisor: Dr. Chih-Wei Yi

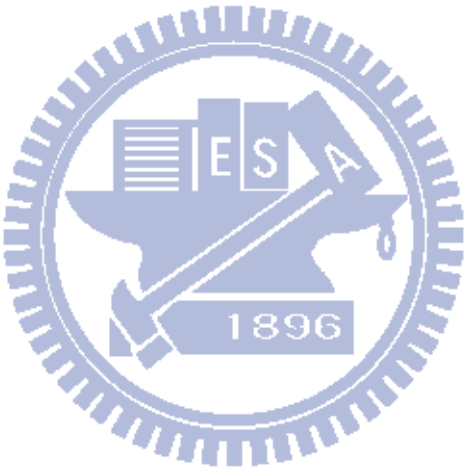
Department (Institute) of Network Engineering
National Chiao Tung University

Abstract

Nowadays, content distribution services comprise most of the Internet traffic, and about half of them is contributed by video streaming. Live streaming (LS), which provides people a way to share what are currently happening in the world, will be one of the main streams of video services on the Internet. Playback delay, the time interval between events occur and the event video are watched by LS users, is the most important metric of LS systems. In this paper, we focus on providing LS services in tree-based P2P networks, so that systems could serve more users under high bandwidth requirement and maximum playback delay constraint. We proposed two algorithms: PCL algorithm, and local optimization algorithm. PCL places newly joining peers in the tree properly so that peers which contribute more upload bandwidth could experience lower playback delay, and local optimization aims to lower the average system playback delay in distributed manner. The simulation result indicates that our algorithms allow systems to serve more users than the system which adopts an algorithm that only considers playback delay. In addition to the simulation work, we have implemented a tree-based P2P LS prototype, which allows us to verify/improve the future proposed algorithms in real Internet.

致 謝

經過兩年的努力，終於完成我的碩士論文。在這期間遭遇到不少研究與人生的難題是我自己無法解決的，必須仰賴許多人的幫助，孝恆欲在此一一致謝。首先，要感謝我的指導教授 易志偉老師，在兩年期間內除了指導我如何分析問題與解決問題外，還需要為我糟糕的英文寫作操心到半夜無法睡覺。更重要的是，老師在生活中給我不少照顧，讓在異鄉的遊子在新竹也能有家的感覺。其次要感謝的是父母親與所有家人，供我成長、讀書，以及培養我從小獨立的個性。雖然父親沒能在我完成學業前就過世了，但我相信父親一定能引我為傲。最後，我要感謝實驗室的學長、同學、學弟妹與所有交通大學的師長與朋友，陪伴我度過這麼有意義的兩年。



目 錄

摘 要 ..... **i**

Abstract ..... **ii**

致 謝 ..... **iii**

目 錄 ..... **iv**

表 目 錄 ..... **vii**

圖 目 錄 ..... **viii**

演 算 法 目 錄 ..... **x**

第一章、緒論 **1**

1.1 動機與目的 2

1.2 文章架構 3

第二章、背景知識 **4**

2.1 主從式架構 V.S. P2P 架構 4

2.1.1 主從式網路架構 4

2.1.2 P2P 架構 6

2.2 P2P 檔案分享 7

2.2.1 Napster – P2P 檔案分享的鼻祖 7

2.2.2 Gnutella 8

2.2.3 BitTorrent 9

2.3 影音串流 10

2.3.1 串流通訊協定 10

2.3.2 緩衝與播放機制 11

2.3.3 影音壓縮格式 12

2.4 隨選視訊服務與即時串流服務 12

2.4.1 隨選視訊服務(VoD) 13

2.4.2 即時串流服務(LS) 13
第三章、相關研究 15
3.1 網路拓撲 15
3.1.1 網狀網路 15
3.1.2 樹狀網路 17
3.1.3 混合型網路 19
3.2 降低樹狀網路的平均 Playback delay 20
第四章、演算法設計 22
4.1 問題模型 23
4.2 父節點候選清單生成演算法 24
4.3 區域最佳化演算法 25
第五章、實驗模擬 31
5.1 實驗環境設定 32
5.2 實驗結果 33
5.3.1 情境一、500Kbps 串流資料率及未設定延遲時間上限值 34
5.3.2 情境二、500Kbps 串流資料率及 200 毫秒延遲時間上限值 38
5.3.3 情境三、1Mbps 串流資料率及未設定延遲時間上限值 42
5.3.4 情境四、1Mbps 串流資料率及 200 毫秒延遲時間上限值 46
第六章、雛型系統設計 50
6.1 系統架構 50
6.1.1 Tracking server 51
6.1.2 Streamer server 51
6.1.3 Peer 52
6.2 軟體模組 52
6.2.1 Tracker 52
6.2.2 Streaming server 54

6.2.3 Peer 56
6.3 系統運行畫面 58
6.3.1 Tracking server 運行畫面 58
6.3.2 Streaming server 運行畫面 58
6.3.3 Peer 運行畫面 59
第七章、結論 60
7.1 討論 60
7.2 未來工作 60
參考文獻 61

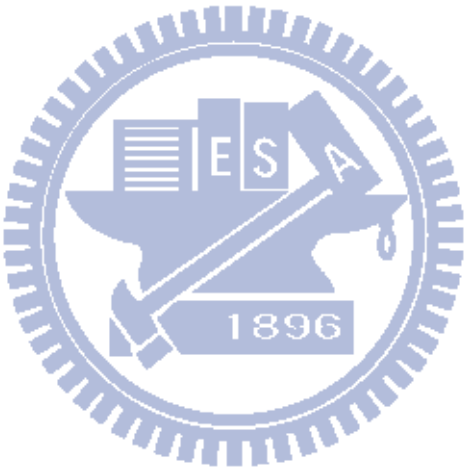


表 目 錄

表 1 點對點延遲時間分佈機率 32
表 2 上載頻寬分佈機率 32
表 3 四種網路模擬情境 33

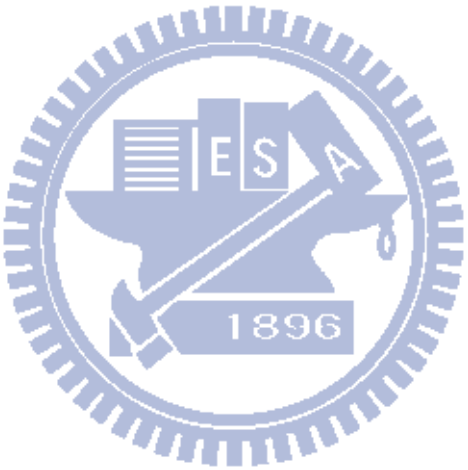


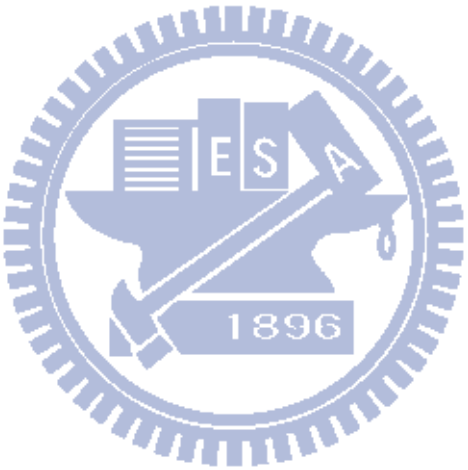
圖 目 錄

圖 1 網際網路流量預測表 2
圖 2 主從式架構 5
圖 3 P2P 架構 7
圖 4 Napster 運作示意圖 8
圖 5 Gnutella 運作示意圖 9
圖 6 網狀網路 16
圖 7 樹狀網路 17
圖 8 SplitStream 串流網路示意圖 18
圖 9 mTreebone 串流網路示意圖 20
圖 10 最佳的節點放置方式 21
圖 11 延遲時間上限值 23
圖 12 區域最佳化 25
圖 13 區域最佳化演算法比較 30
圖 14 情境一之線上節點個數 35
圖 15 情境一之系統平均 Playback delay 35
圖 16 情境一中 Greedy 的平均 Playback delay 36
圖 17 情境一中 PCL 的平均 Playback delay 36
圖 18 情境一中 OPT 的平均 Playback delay 37
圖 19 情境二之線上節點個數 39
圖 20 情境二之系統平均 Playback delay 39
圖 21 情境二中 Greedy 的平均 Playback delay 40
圖 22 情境二中 PCL 的平均 Playback delay 40
圖 23 情境二中 OPT 的平均 Playback delay 41

圖 24 情境三之線上節點個數 43
圖 25 情境三之系統平均 Playback delay 43
圖 26 情境三中 Greedy 的平均 Playback delay 44
圖 27 情境三中 PCL 的平均 Playback delay 44
圖 28 情境三中 OPT 的平均 Playback delay 45
圖 29 情境四之線上節點個數 47
圖 30 情境四之系統平均 Playback delay 47
圖 31 情境四中 Greedy 的平均 Playback delay 48
圖 32 情境四中 PCL 的平均 Playback delay 48
圖 33 情境四中 OPT 的平均 Playback delay 49
圖 34 雛型系統架構圖 51
圖 35 Tracking server 的軟體模組圖 52
圖 36 Channel manager 的軟體模組圖 54
圖 37 Streaming server 的軟體模組圖 55
圖 38 Peer 的軟體模組圖 56
圖 39 Video player 軟體模組圖 57
圖 40 Topology monitor 的運行畫面 58
圖 41 Streaming server 運行畫面 59
圖 42 Peer 的運行畫面 59

演 算 法 目 錄

演算法 1 父節點候選清單生成演算法 24
演算法 2 Random 演算法 26
演算法 3 Largest upload bandwidth 演算法 27
演算法 4 Lowest playback delay 演算法 28
演算法 5 Optimal 演算法 29
演算法 6 Greedy 演算法 31



第一章、緒論

隨著網際網路的發展，透過網際網路提供多樣化的網路服務已成為現今最熱門的應用領域。內容分發服務(Content distribution services, CDSs)是常見的網路服務之一，這類型的服務提供使用者一個將內容與他人分享的平台，例如：使用者可以瀏覽別人撰寫的網頁內容、下載別人分享的檔案，或是觀看別人提供的影音串流。

由於內容分發服務的成功，網際網路的流量正在大幅的成長中。根據 Cisco Visual Networking Index[1]的預測，網際網路的總流量在 2014 年時將超過 2009 年的 4 倍；如圖 1 所示，到了 2014 年，網頁瀏覽、檔案分享及影音串流服務的總流量則分別佔有總流量的 15%、27%及 56%。此外，由於用戶端網路頻寬的快速成長，那些過去不易實現的網路服務如今可嶄露頭角，如高畫質影音串流服務(High-definition Live Streaming, HDLS)就是其中最顯著的例子。根據[1]的預測，高畫質影音服務的流量將在 2012 年超過普通畫質(Standard-definition, SD)影音服務的流量，並佔有 2014 年總影音流量的 46%。

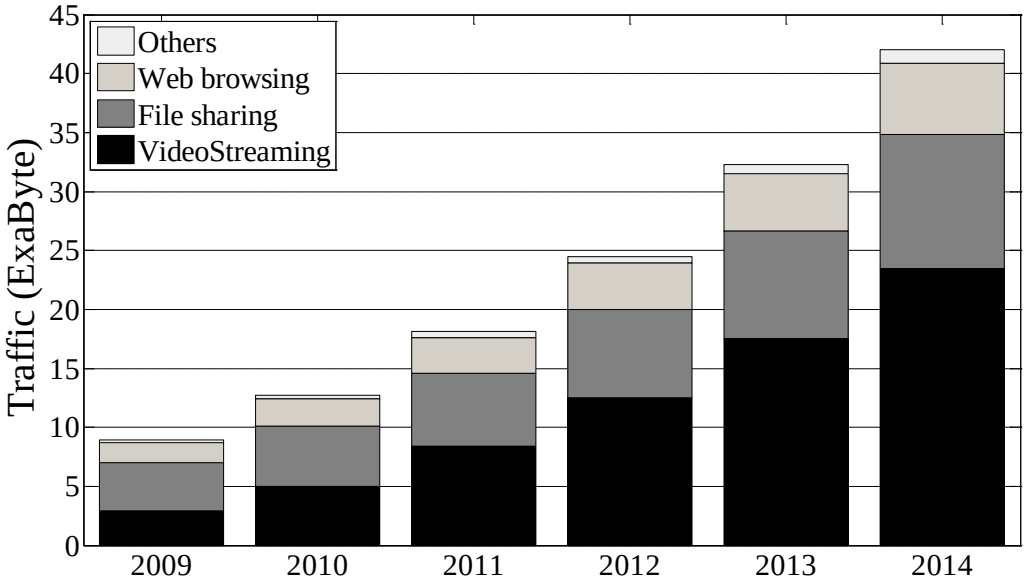


圖 1 網際網路流量預測表

1.1 動機與目的

影音串流是透過網際網路分享影音內容的主要方式之一。隨著影音串流、壓縮技術的發展及用戶端網路頻寬的快速成長，越來越多的影音串流服務提供高畫質的影音來源。此外，未來人們將以高畫質即時影音串流(HDLS)來取代傳統的語音電話，屆時將需要更多的網路頻寬需求。點對點架構(P2P Architecture)，提供比主從式架構(Client/Server Architecture)較佳的系統擴展性(Scalability)與容錯性(Fault tolerant)，並能夠充分利用 P2P 網路中用戶端節點的網路頻寬。因此 P2P 架構特別適合於提供 HDLS 服務。

播放延遲時間(Playback delay)，是評量一個即時影音串流系統好壞的主要依據。播放延遲時間的定義為自串流伺服器產生一段影音內容開始，直到這段影音內容被使用者端接收到的這段時間區間。若即時影音串流系統無法在有效時間內將影音內容傳遞到使用者端，不但失去了即時串流的意義，也降低了透過即時串流達到雙向互動的可能性。因此，系統中使用者所經歷的平均播放延遲時間在某總程度上能夠代表即時串流系統的系統容量。我們定義即時串流系統的系統容量為：在系統平均播放延遲時間低於一個系統所設定的延遲時間上限值(Maximum delay constraint)時，系統最多能夠服務多少使用

者。

本篇文章將探討如何在 P2P 網路中提供 HDLS 服務。為了達到目的，除了須整合 P2P 網路中節點的網路運行狀況異質性外(如網路頻寬及點對點延遲時間)，尚須考慮如何提升 HDLS 系統的容量。我們提出了兩個利用樹狀 P2P 網路提供 HDLS 服務的演算法，期望在高網路頻寬需求下透過降低系統平均播放延遲時間來達到提升系統容量的目的，並在透過模擬實驗來驗證兩個演算法的可行性。此外，我們實作了一個樹狀 P2P 網路的即時影音串流系統雛型，該雛型系統提供後續的研究一個真實的網路實測平台。

1.2 文章架構

在第二章中，我們先介紹相關背景知識，其中包含主從式架構與 P2P 架構的比較，及介紹兩個 P2P 技術的熱門應用，分別是檔案分享與影音串流。在第三章中，我們分析兩種常被用來建構 P2P 即時串流系統的兩種網路型態，分別是網狀網路(Mesh-based)及樹狀網路(Tree-based)。在第四章中，我們提出了兩個演算法來提升樹狀 P2P 即時影音串流系統的容量，並將模擬實驗結果呈現在第五章。在第六章中，我們介紹一個即時影音串流系統的雛型設計。最後，我們在第七章給予關於本篇文章的結論。

第二章、背景知識

2.1 主從式架構 V.S. P2P 架構

享受 CDSs 已成為人們生活的一部分。人們在 Twitter 及 Facebook 等社群網站上與朋友分享近況、將旅行時所拍的照片上傳到 Picasa 及 Flickr 等網路相簿、透過 Google 及 Yahoo!等網路搜尋引擎尋找網站、在 Wikipedia 及 Answers.com 等網站上吸收新穎的知識，或是在 YouTube 及 Justin.tv 等網站上觀看影音串流。可想而知的是，上述這些熱門的 CDSs 必定佈建了強健的系統網路來應付來自世界各地使用者每日數以萬計的服務請求。

2.1.1 主從式網路架構

大多數的 CDSs 佈建了配備有多台伺服器的機房，並在伺服器間套用了負載平衡(Load balancing)的機制，使得處理服務的工作量能被分散到所有的伺服器。如圖 2所示，用戶端將請求服務的訊息送至伺服器，例如查詢資料庫、檔案下載或是其他複雜的運算工作。伺服器端可能由多台伺服器組成，用來完成不同的任務或是分散工作負載量。待伺服器處理完請求服務後，會將計算結果回覆給用戶端。用戶端只需專注在如何將結果有效率的的呈現給使用者即可。如此簡單的設計概念從 1950 年代的大型電腦(Mainframe)開始，直至今日的資料中心(Data center)都還在使用。甚至，現今常見的網路協定，如 HTTP、FTP、SMTP 及 DNS 也都是基於主從式架構的概念所設計的。然而，缺乏安全性及擴展性是主從式架構兩個主要的缺點。若伺服器遭受到分散式阻斷(Distributed Denial-of-Service, DDoS)等網路攻擊，可能導致整個 CDSs 的停擺。此外，由於伺服器

的計算資源與網路頻寬有限，使得系統容量受到限制，這對於 CDSs 的推廣有極大的影響。諸多研究在探討如何改善主從式架構的安全性與擴展性，如[2]、[3]。不幸的是，不管是提升安全性或是增進擴展性皆需要花費昂貴的硬體與維護成本。

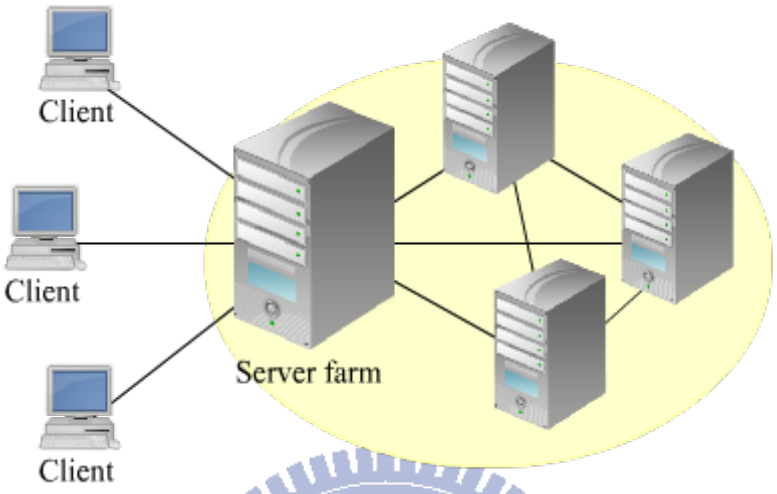


圖 2 主從式架構

範例系統 1: Akamai

Akamai[4]是著名的內容分發網路(Content Distribution Network, CDN)服務提供商。其在世界各地佈建分散式平台，幫助網站內容提供商(Content provider)將內容有效率的呈現給瀏覽網站的使用者。Content provider 先將欲呈現給使用者的內容提供給 Akamai，然後在自己維護的網頁中呈現由 Akamai 伺服器下載的網站內容。藉此，Content provider 可無須擔心伺服器擴展性的問題。諸多為人熟知的內容提供商都使用 Akamai 的服務，例如英國國家廣播公司 BBC、Hulu、中國中央電視台 CCTV，甚至連美國白宮都透過 Akamai 來提供影音串流服務。

Akamai 在世界各地佈建了包含上千台伺服器的 Akamai network，且伺服器的數量仍正在依需求增加中。Akamai 會依據瀏覽網站的使用者的真實地理所在地，以最接近使用者、或是具有最佳連線品質的伺服器來服務使用者。此外，不同型態的內容，例如文字、影像、或是影音，都可能由不同的伺服器來負責服務。Akamai 期望能透過上述的機制來提供瀏覽網站使用者有效率的上網環境。根據[5]，Akamai 每日服務來自世界

各地數十億的內容請求。在 2010 年第四季內，總計有 556 百萬個相異的 IP 位址使用到 Akamai 服務，這個數量比 2009 年第四季多了 20 個百分比。

範例系統 2: Dropbox

Dropbox[6]是一個線上檔案同步與備份系統，他允許使用者將檔案儲存在 Dropbox 網路上，且那些檔案會被及時的同步到使用者的其他裝置上。透過邀請的機制，多個使用者可以共同分享及時同步的資料夾。被修改的檔案都會被及時同步給所有一同分享的人。Dropbox 會自動備份檔案，並為每個檔案提供一個月期限的版本控制服務。在檔案的一個月生命週期內，使用者可以將檔案回溯至較舊的版本、或是當檔案遺失時將他們重置。此外，Dropbox 提供了一個方便使用者公開分享檔案給其他未安裝 Dropbox 軟體的使用者的管道。透過 HTTP 服務，未安裝 Dropbox 軟體的使用者可以直接下載被公開分享的檔案。Dropbox 目前仍正在快速成長中。其在 2007 年成立，在 2009 年時已有二百萬用戶數，2010 年時用戶數成長一倍，達到四百萬人。直至今日為止，已有大於二千五百萬的用戶數。

2.1.2 P2P 架構

P2P 架構是專門針對主從式架構擴展性不足而提出的解決方案。P2P 網路透過一個建構在應用層(Application layer)的重疊網路(Overlay network)將所有用戶端節點連接起來，如圖 3 所示。若兩個節點間有直接的網路連線，則他們彼此互相稱為鄰居(Neighbor)。鄰居間可以互相提供服務，而不是只仰賴於特定的伺服器。這樣的服務方式將處理服務請求的工作量分散到 P2P 網路中所有的節點。因此，可將 P2P 網路視為一個橫跨網際網路的機房，且與生俱有平衡負載的能力。藉此，可節省伺服器的計算資源及妥善利用網路頻寬來服務更多的節點。然而，P2P 架構仍然有安全性的問題，且可能比主從式架構更加棘手。儘管在 P2P 網路中沒有可能造成服務中斷的關鍵性角色，但具有惡意的使用者可能透過不正常的節點行為來試圖破壞 P2P 網路服務，如 Sybil attack 及 Pollution attack 即是。相關的研究如[7]、[8]就是為了解決上述的問題。

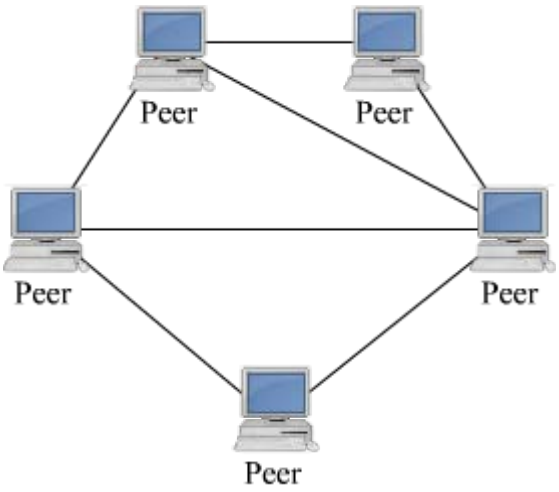


圖 3 P2P 架構

2.2 P2P 檔案分享

2.2.1 Napster – P2P 檔案分享的鼻祖

Napster 是最早採用 P2P 架構的檔案分享應用程式。使用者可以在 Napster 平台中彼此分享 MP3 檔案。Napster 使用中央式的目錄伺服器來記錄目前線上的使用者資訊及他們欲分享的 MP3 檔案清單。使用者可利用關鍵字在目錄伺服器上搜尋其他人分享的檔案，目錄伺服器會將符合搜尋結果的檔案資訊(Metadata)回覆給使用者。取得檔案資訊後，使用者便可直接與檔案分享者建立直接連線並下載 MP3 檔案。如圖 4 所示，Peer A 先從 Directory server 得知可從 Peer B 下載到檔案 music.mp3。然後，Peer A 直接和 Peer B 建立連線並下載 music.mp3。下載完成後，Peer A 會通知 Directory server，告知 Directory server 說 Peer A 也可提供其他使用者下載 music.mp3。

Napster 服務運作在 1999 年 6 月到 2001 年 7 月之間，總計有 8000 萬個使用者帳戶，最多同時有 2500 萬個檔案被分享。不幸的是，Napster 最終因為中央式目錄伺服器所引發的 MP3 版權爭議而中止提供服務。

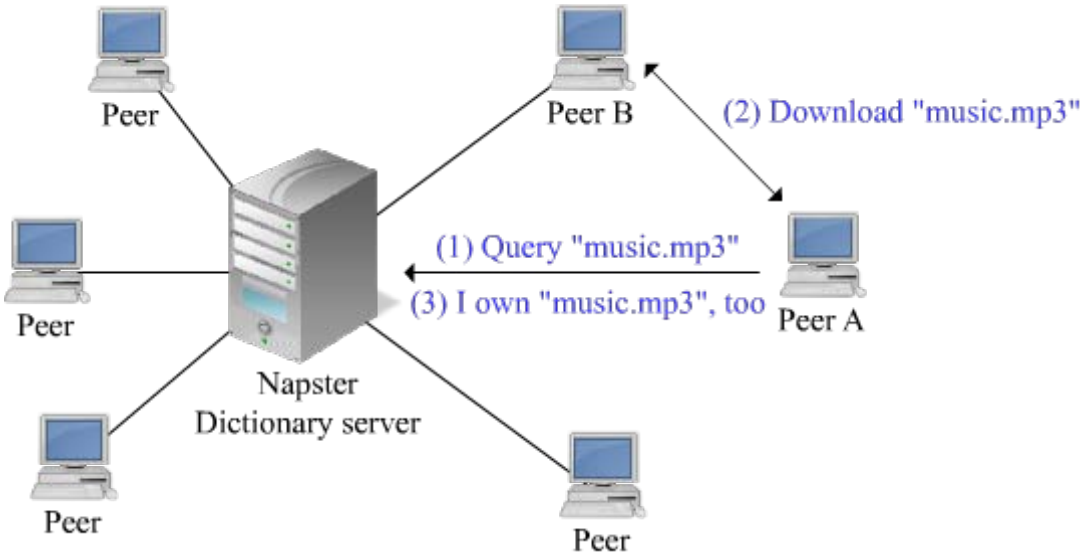


圖 4 Napster 運作示意圖

2.2.2 Gnutella

有鑑於 Napster 引發的版權爭議，後 Napster 時代的檔案分享應用程式在設計規劃時常採用純 P2P(Pure P2P)架構來規避法律議題。為了避免使用中央伺服器，P2P 節點必須透過分散式的方法來形成 P2P 網路，而 Gnutella[9]無疑是最經典的例子了。在 Gnutella 中，新加入的節點必須先連線上一個鄰居節點並取得部分的 P2P 節點清單。Gnutella 設計了一套分散式的方法來達到如同 Napster 查詢目錄的目的。節點使用 Controlled Flooding 或是 Random walking 的方式將關鍵字搜尋訊息透過鄰居節點散佈到網路中，直到找到符合關鍵字配對的檔案為止。如圖 5 所示，Peer A 使用 Time-to-Live (TTL)為 2 的 Controlled Flooding，將搜尋訊息透過鄰居節點散佈出去。由於 Peer B 擁有檔案 music.mp3，因此 Peer A 可以從 Peer B 直接下載檔案。

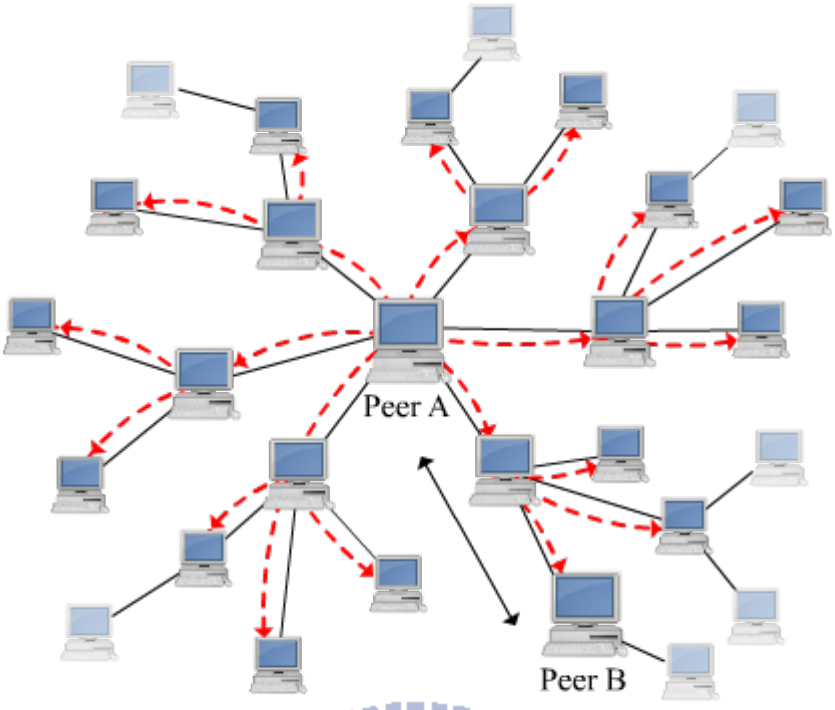


圖 5 Gnutella 運作示意圖

2.2.3 BitTorrent

BitTorrent[10]是現今最熱門的 P2P 檔案分享服務之一。人們透過實作 BitTorrent 協定的用戶端程式成為 BitTorrent 網路中的一個節點，並與其他需要相同檔案的節點一起分享與下載檔案。被分享的檔案會被切割成多個區塊(Piece)，而擁有所有 Piece 的節點被稱為種子節點(Seeder)。其他節點可以從種子節點將部分的 Piece 下載回來，並與其他節點分享自己擁有的 Piece。檔案的最初分享人必須產生一個 Torrent 檔案，並將這個檔案以任何方式散佈給其他需要該檔案的使用者，如 E-mail、論壇等。Torrent 檔案中描述了關於被分享檔案的敘述，例如檔案大小及用來加解密的文字字串。此外，Torrent 檔案中必須指定一個追蹤伺服器(Tracker)。其他需要該檔案的使用者必須先取得該 Torrent 檔案，並連線到所指定的 Tracker。Tracker 會告訴使用者網路中其他正在下載該分享檔案的節點的 IP 位址。藉此，正在下載相同檔案的節點們就能夠聚在一起互相交換自己所擁有的 Piece，進而將所下載的 Piece 拼湊成完整的檔案。

根據 ipoque[11]的研究，在 2008-2009 年間，56%的網際網路流量來自於 P2P 應用程式，且 60%的 P2P 應用流量來自於 BitTorrent 檔案分享服務。

2.3 影音串流

影音串流是透過網際網路分享影音內容的主要方式之一。串流伺服器持續的將影音內容傳送至使用者端。使用者端則一邊接收、一邊播放所接收到的影音內容，並不需要將影音檔案完整下載。

2.3.1 串流通訊協定

不同的影音串流服務可能使用不同的串流通訊協定。使用者端必須依據串流伺服器的不同，以串流伺服器所支援的串流通訊協定，才能順利地從串流伺服器溝通，並取得影音內容。

1. HTTP Streaming

HTTP 是最常被使用的串流通訊協定。將影音檔案放置在 Web 伺服器中，使用者端利用網頁瀏覽器就可即時地從 Web 伺服器下載影音串流回來觀看。其優點在於 HTTP 封包較不易受到防火牆及代理伺服器的干擾，而缺點是 HTTP 須透過 TCP 的可靠傳輸。儘管可靠傳輸帶來較佳的影音品質，但並不適用於即時影音串流等即時導向的應用。

2. RTP/RTCP/RTSP

Real-Time Transmission Protocol(RTP)是另一個常用的串流通訊協定。RTP 定義了標準的封包格式來裝載影音內容，且透過 UDP 通訊協定來傳輸。Real-Time Control Protocol(RTCP)及 Real-Time Streaming Protocol(RTSP)常被用來與 RTP 搭配的輔助通訊協定。RTCP 可針對一個 RTP 連線提供統計與同步的服務，RTSP 則提供使用者端控制影音播放的功能，例如暫停、快轉、及倒帶等。由於 UDP 的傳輸

較適合即時的應用，且 RTSP 能提供 RTP 連線兩端互動的可能，因此 RTP 常被用來提供影音串流服務。然而，除了須架設支援上述串流通訊協定的串流伺服器外，在多數的商用網路中，會基於安全性的考量而使用防火牆及代理伺服器將 UDP 封包隔絕，使得無法達到影音串流的目的。

3. P2P Protocol

透過 P2P 網路提供影音串流服務也是常見的串流方法。節點間使用自行定義的通訊協定來交換影音內容。一般來說，節點間交換影音內容的方式分為 Pull 或是 Push。若是採用 Pull，鄰居節點之間彼此須知道對方擁有哪些影音內容。若欲從某鄰居節點取得自己未有的影音內容，節點須送出影音內容請求至該鄰居節點，並下載所請求的影音內容。反之，若是採用 Push，影音內容會依據網路拓撲來散佈到所有節點。例如，在一個樹狀網路中，節點須將從父節點傳遞過來的影音內容轉送到自己的所有子節點。

2.3.2 緩衝與播放機制

Internet Protocol(IP)是現今網際網路主要使用的網路層通訊協定，從 1980 年代一直使用至今。分封交換(Packet switching)是 IP 重要的觀念之一。IP 封包從傳送端送出後，須經過一連串的路由器的轉送，才能夠到達原先預定的目的地。然而，分封交換機制並不利於影音串流服務的發展，我們列出兩個主要原因。

1. 路徑多樣性 (Path diversity)

路由器會依據路由演算法，選擇一個最佳的轉送目標(Next hop)來轉送封包。由於網路的狀況是具有變動性的，在不同時間點，相同的路由器可能將相同的封包轉送到不同的目標，此特性造就了封包的路徑多樣性。因此，若在 IP 中提供影音串流服務，可能導致一連串的影音內容無法依序且穩定的到達目的地，形成播放錯亂的情形。

2. 有限的路由器資源

路由器的資源有限，例如計算能力不足或是路由佇列的長度是有限的。因此，若路由器資源不足時，必須將部分待處理的封包丟棄(Drop)。若 IP 所服務的對象是可靠傳輸通訊協定(Reliable Transmission Protocol)，如 TCP，一旦有封包被路由器丟棄，則封包重送機制會被啟動，目的是要確保所有封包都能依序且正確的被傳送到目的地。因此，若在 IP 中提供可靠的影音串流服務，封包重送機制將導致影音內容到達目的地的時間被延遲。

上述兩點皆為 IP 不利於提供影音串流服務的因素。由於 IP 是現今網路的主流通訊協定，且期望修改 IP 是不切實際的，因此影音串流服務必須想到妥協的方式。其中一個妥協的方式為在使用者端採用搭載播放緩衝區(Playback buffer)的串流播放器，讓播放緩衝區先暫存一段時間的影音內容，直到所暫存的數量足以應付上述兩個問題後才開始播放影音內容。

2.3.3 影音壓縮格式

由於影音資料量龐大，串流伺服器不可能直接將未壓縮過的影音內容透過網際網路傳遞給使用者。透過有效率的影音壓縮演算法，可大幅減少影音檔案中的冗餘資料，並只些微影響到影音內容的視覺品質。更重要的是，透過壓縮影音檔案，可降低影音串流伺服器的頻寬使用量，使得串流伺服器能夠服務更多的使用者。H.264/AVC 或稱 MPEG4 Part 10，是現今影音壓縮的主流演算法。與其他先前的演算法相比較，H.264/AVC 能夠以較低的資料量提供相同的影音品質。因此，H.264/AVC 已被廣泛的應用在影音串流服務上，由其是 HDLS。

2.4 隨選視訊服務與即時串流服務

隨選視訊(Video-on-Demand, VoD)和即時串流(Live streaming, LS)是兩種不同型式的影音串流服務。

2.4.1 隨選視訊服務(VoD)

VoD 服務讓使用者可以完全控制影音串流的播放，例如，暫停、快轉及倒帶等功能。在同一時間，對於相同的影片，不同的使用者所觀看到的影音內容可能是不同的。因此，若欲提供 VoD 服務，最直覺的方法就是建置一台 VoD 伺服器，為不同的使用者提供獨立的影音串流服務，如[13]即是。

在 VoD 服務中，若想要達到控制播放的目的，又想要讓影音串流能夠順暢播放，使用者端必須準備極大的播放緩衝區，用來暫存過去已被播放過的以及預先提取(Prefetch)為了後續播放的影音內容。事實上，這樣的緩衝機制非常有利於在 P2P 網路中提供 VoD 服務。若每個節點都能準備較大的播放緩衝區，將可以提升節點間交換影音內容的可能性。諸多研究在探討如何在 P2P 網路中提供 VoD 服務，如[14]、[15]即是。甚至，市面上已有許多商業軟體提供 VoD 服務，如 PPStream[16]、PPTV[17]即是。

2.4.2 即時串流服務(LS)

LS服務和VoD服務是本質上全然不同的兩種影音串流服務。在LS服務中，使用者所觀看到的影音內容是串流伺服器正在同步廣播的。相較於VoD服務，LS服務的使用者端無須準備如VoD服務般大的播放緩衝區。原因是無法預先提取後續的影音內容，而且保留已被播放過的影音內容是沒有意義的。由於影音串流是即時廣播的，LS服務是否提供控制影音串流播放的功能，並不是主要關注的議題。反之，LS服務最關注的是播放延遲時間是否太長。若播放延遲時間過長，就失去即時廣播的意義。例如，電視台正在網路實況轉播一場棒球比賽，若打者擊出逆轉勝全壘打的影像過了30秒後才能被傳達到觀眾眼中，將可能大幅降低觀眾透過網路觀看比賽的興致。

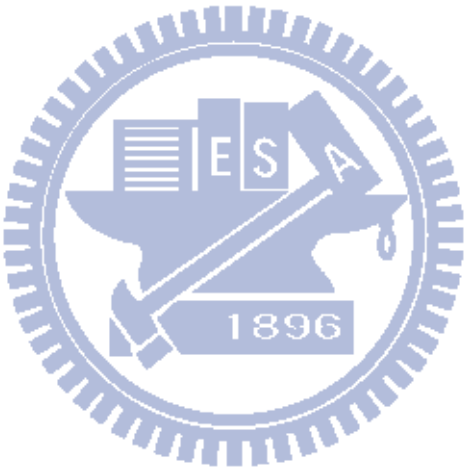
網路頻寬是影響播放延遲時間的重要因素。若網路中存在一些網路連線的頻寬低於影音串流服務的頻寬需求，會導致影音內容無法準時被送達使用者端。對於多數的網路連線，其下載頻寬遠大於上載頻寬，如ADSL即是。在此篇文章中，我們假設網路連線

的頻寬瓶頸在於其上載頻寬，透過改善系統網路的上載頻寬可降低影音串流的播放延遲時間。然而，若是透過P2P網路來提供LS服務，有兩個衍生的問題。

1. 節點須貢獻自己的上載頻寬來和其他節點交換影音內容。

2. 播放延遲時間是每次交換動作所花時間的總和。

因此，如何在P2P網路中設計影音內容交換演算法，來兼顧節點的上載頻寬異質性與降低播放延遲時間，是設計P2P LS服務的一大挑戰。



第三章、相關研究

在上一章中，我們提到播放延遲時間的長短是衡量一個 LS 系統好壞的重要依據，而提供 P2P LS 服務更是一大挑戰。在本章中，我們將介紹兩種常被用來建構 P2P LS 系統的兩種網路型態，分別是網狀網路(Mesh-based)及樹狀網路(Tree-based)，並介紹如何依據這兩種網路型態各自的優點，將他們整合成混合型網路(Hybrid)型式。最後，我們介紹如何在樹狀網路中降低系統平均播放延遲時間。

3.1 網路拓撲

3.1.1 網狀網路

在網狀網路中，節點是隨機散佈在網路中的。欲加入網路的節點會先從初始伺服器(Bootstrap server)或是某些節點取得目前網路中的部分節點清單，並從清單中選擇部分節點作為鄰居，如圖 6 所示。鄰居節點必須定期的交換緩衝區分佈表(Buffer map)給對方，用來告知對方自己擁有哪些影音區塊(Piece)。如此，節點可以知道自己沒有哪些影音 Piece 而其鄰居有。在網狀網路中，鄰居間主要使用 Pull 的方式來交換影音 Piece，節點須先送出區塊內容請求(Piece request)給鄰居，然後才能將想要的影音 Piece 從鄰居那邊下載回來。然而，想要的影音 Piece 可能同時被多個鄰居所擁有，因此必須設計一個鄰居挑選演算法(Neighbor selection algorithm)，讓節點能夠從多個鄰居中挑選其一來送出 Piece request。該演算法必須基於降低使用者的播放延遲時間的原則來設計。例如，若影音 Piece 能夠在限定的時間內從其中一個鄰居下載回來，而從其他鄰居下載卻不行，則那個鄰居當然就會被挑選出來。要實現上述這種想法可以利用節點的真實地理位置資訊來達成，每次挑選鄰居時，皆挑選和自己有最長相同 IP Prefix 的鄰居即可。因為地理

位置越相近的兩個節點其 IP Prefix 會越長，IP Prefix 越長則代表兩個節點間的點對點延遲時間應該越短。目前已有諸多相關的研究，如[18]、[19]即是。

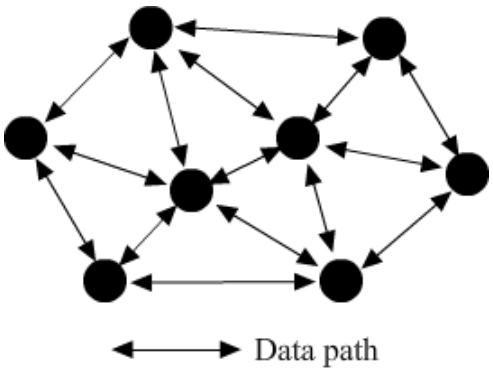


圖 6 網狀網路

然而，由於節點必須先利用鄰居挑選演算法挑選出鄰居，並送出 Piece request 至該鄰居才能後才能開始下載影音 Piece 的工作，這使得採用 Pull 方式的效率遠不及採用 Push 的方式。在 Push 的方式中，節點無須送 Piece request 至鄰居，也因此沒有鄰居挑選演算法的需求。取而代之的是，節點會將所收到的影音 Piece 即時轉送給幾個事先決定好的鄰居。由於減少了鄰居間協調的麻煩，Push 的方式在效率上能表現的比 Pull 的方式好。但若沒有妥善決定節點應轉送資料的鄰居，會造成相同的 Piece 從不同的鄰居被送到相同的節點，導致網路頻寬被嚴重的浪費。

CoolStreaming/DONet[20]是最典型在網狀網路上以 Pull 的方式提供 LS 服務的系統。在 CoolStreaming/DONet 中，節點必須定期與鄰居交換 Buffer map，並從鄰居那邊下載自己需要的影音 Piece。CoolStreaming/DONet 由三個主要軟體模組所組成，分別是：

1. Membership manager

由於系統中沒有中央伺服器，CoolStreaming/DONet 實作 SCAMP[21]來散佈系統控制訊息給網路上所有的節點。SCAMP 是一個 Gossip-based 的協定，每個節點都只需要知道網路上部分節點的資訊就能夠以分散式的方式將控制資訊散佈給網路上所有的節點。

2. Partnership manager

每個節點都須動態維護一個穩定數量的鄰居清單。

3. Data scheduler

以資料排程演算法決定哪些影音 Piece 應該從哪些鄰居那邊下載回來。

3.1.2 樹狀網路

在樹狀網路中，節點間連接成一棵以串流伺服器為樹根的有向樹，且節點間主要以 Push 的方式交換影音 Piece。如圖 7 所示，串流伺服器將所產生的影音 Piece 轉送給所有自己的子節點，同樣地，節點接收到來自父節點的影音 Piece 後會將他們再轉送給自己的所有子節點。藉此，串流伺服器所產生的影音 Piece 就能夠被如此簡單的方式散佈到網路上所有的節點。諸多研究都使以類似的方式來達到 P2P LS 的目的，如 Overcasr[22]、ZIGZAG[23]及 Climber[24]即是。

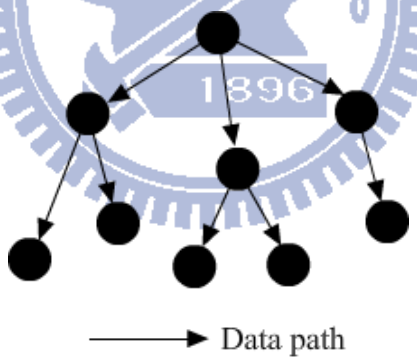


圖 7 樹狀網路

然而，樹狀網路有一些明顯的缺點，其中之一是沒有妥善利用所有節點的上載頻寬。在樹狀網路底部的節點(或稱葉子節點)只有下載影音 Piece，卻沒有貢獻半點他們的上載頻寬給串流系統，這對於其他有貢獻上載頻寬的節點(內部節點)來說是不公平的。一旦樹狀網路的規模越來越大，葉子節點的成長速度將遠快於內部節點的成長速度，屆時將使得即時影音串流系統越來越不平衡。另一個明顯的缺點是，在劇烈變動的節點行為下(如加入、離開)，樹狀網路的拓撲結構會被嚴重破壞，尤其是越接近樹根的節點離開網

路時。一旦樹狀網路被破壞，網路中的節點就必須被重整，屆時將花費不少用來修復樹狀網路拓撲的時間成本。

諸多研究在探討如何解決上述的問題，其中一種解決方法嘗試將節點組織成多棵互相重疊的樹，如 SplitStream[25]及 CoopNet[26]。舉 SplitStream 為例，SplitStream 將每個 Piece 再切割成 K 個子區段(Stripe)，並以 K 棵不同的樹來散佈這 K 個 Stripe。在最理想的情況下，SplitStream 期望將所有節點組織成多棵 Internal-Node-Disjoint-Trees。意思是每個節點會重複出現在 K 棵樹中，但只有在一棵樹中是內部節點，在其餘樹中都是葉子節點。如圖 8 所示，網路中包含兩棵樹，SubTree 1 及 SubTree 2，分別用來散佈 Stripe 1 及 Stripe 2。節點 1、2、3 在 SubTree 1 中是內部節點，但在 SubTree 2 中卻是葉子節點。節點 4、5、7 在 SubTree 2 中是內部節點，但在 SubTree 1 中卻是葉子節點。唯有節點 6 在兩棵樹中都是葉子節點。由於所有節點都至少是一個內部節點，因此他們都必須貢獻自己的上載頻寬將所收到的子區段轉送給自己的所有子節點。藉此，除了能妥善利用節點上載頻寬外，Internal-Node-Disjoint-Trees 的設計對於影音的播放也有幫助：由於每個節點只會在一棵樹中是內部節點，因此，某個節點離開網路只會暫時影響到相對應的 Stripe 而已。

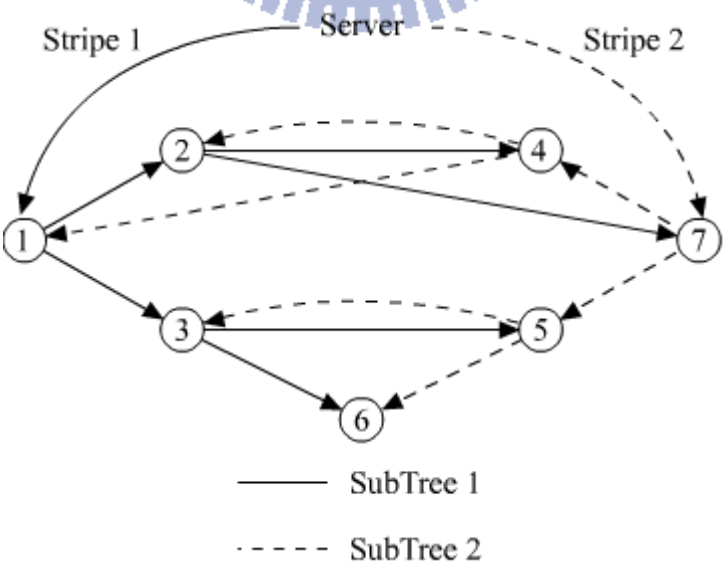


圖 8 SplitStream 串流網路示意圖

3.1.3 混合型網路

在樹狀網路中，由於其節省不必要的控制訊號傳遞，影音 Piece 可以被有效率的沿著樹狀拓撲散佈到所有節點。然而，在劇烈變動的節點行為下(如加入、離開)，樹狀網路的拓撲結構會被嚴重破壞，尤其是越接近樹根的節點離開網路時。一旦樹狀網路被破壞，網路中的節點就必須被重整，屆時將花費不少用來修復樹狀網路拓撲的時間成本。如先前提到的，不少研究在探討如何以建構多棵樹的方式來解決單一棵樹的問題，如 SplitStream 所提出的 Internal-Node-Disjoint-Trees 概念。但事實上，這樣的概念不但難以設計也不易實現。在網狀網路中，節點間主要採用 Pull 的方式來交換影音 Piece，因此相較於樹狀網路能夠提供較佳的網路拓撲容錯性，因為單一節點離開該網路並不會對串流服務造成太大的影響。然而，Pull 的方式需要傳遞大量的控制訊號與資料排程，且不易估計節點的播放延遲時間。

諸多研究如 mTreebone[27]，整合了樹狀網路與網狀網路的優點，提出了針對上述問題的解決方案。如圖 9 所示，mTreebone 提出了樹骨(Treebone)的概念，影音 Piece 在正常的情況下是透過樹骨 Push 給網路上的所有節點。樹骨是由停留在網路中較久的節點所組成，原因是樹骨中的成員必須為系統提供穩定的服務。除了樹骨之外，所有節點另外被一個網狀網路相互連接起來。每當樹骨中的成員離開了網路，會造成部分的節點暫時無法接收到影音 Piece，這時候受影響的節點就會利用 Pull 的方式從網狀網路中的鄰居取得影音 Piece。藉此，mTreebone 融合了樹狀網路和網狀網路的優點，如樹狀網路能有效率的傳的影音 Piece、網狀網路的拓撲容錯性及妥善利用節點上載頻寬，且相較於多棵樹的解決方案容易許多。

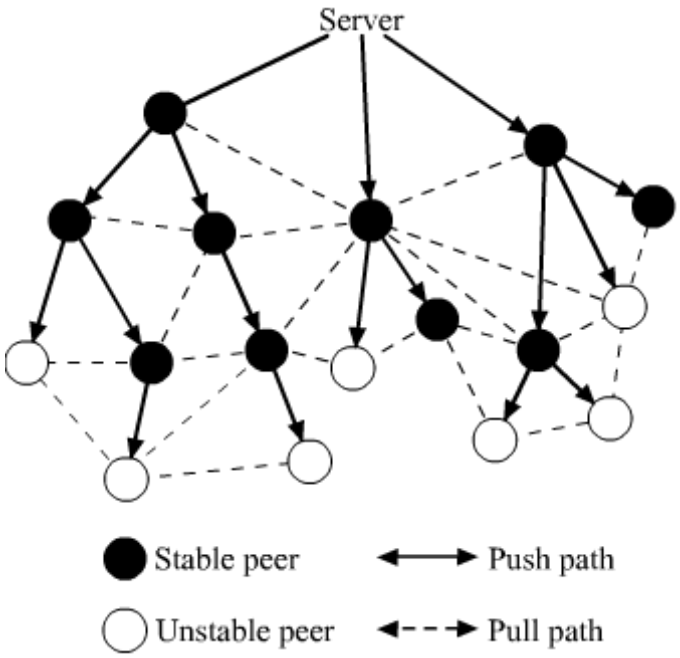


圖 9 mTreebone 串流網路示意圖

在接下來的文章中，我們將專注於透過樹狀網路來提供 P2P LS 服務。因為除了有效率、易於設計與實作外，也易於套用階層式的統計與管理機制。更重要的是，樹狀網路有助於計算節點的播放延遲時間，這對於著重在降低播放延遲時間的研究是良好的研究平台。

3.2 降低樹狀網路的平均播放延遲時間

在樹狀網路中，要計算節點的播放延遲時間，只需要將影音 Piece 從樹根傳遞到節點期間所花費的所有延遲時間加總起來即可。因此，藉由降低影音 Piece 從樹根到節點的傳遞路徑中被轉送的次數，可以達到降低系統平均播放延遲時間的目的。根據諸多研究如[28]、[29]發現，將具有較大上載頻寬的節點放置在距離樹根較近的位置，能夠有效降低系統平均播放延遲時間。在[28]中，作者以數學證明的方式證明論述的正確性，而在[29]中，作者則以模擬的方式來驗證。作者假設網路中有 1000 個節點，且依據上載頻寬被分為三類，其中 100 個節點有 1Mbps 的上載頻寬、300 個節點有 400Kbps 的上載頻寬，剩餘 600 個節點有 200Kbps 的上載頻寬。串流伺服器具有 1Mbps 的上載頻寬且提供資料率為 100Kbps 的影音串流。作者期望透過窮舉的方式找出具有最低系統平均播

放延遲時間的樹狀網路拓撲，其結果如圖 10 所示，在具有最低系統平均播放延遲時間的樹狀網路拓撲中，具 1Mbps 上載頻寬的節點都被放置在樹狀網路中的第一層及第二層，具 400Kbps 上載頻寬的節點大多被放置在樹狀網路中的第三層，而具 200Kbps 上載頻寬的節點則大多被放置在樹狀網路的底部。

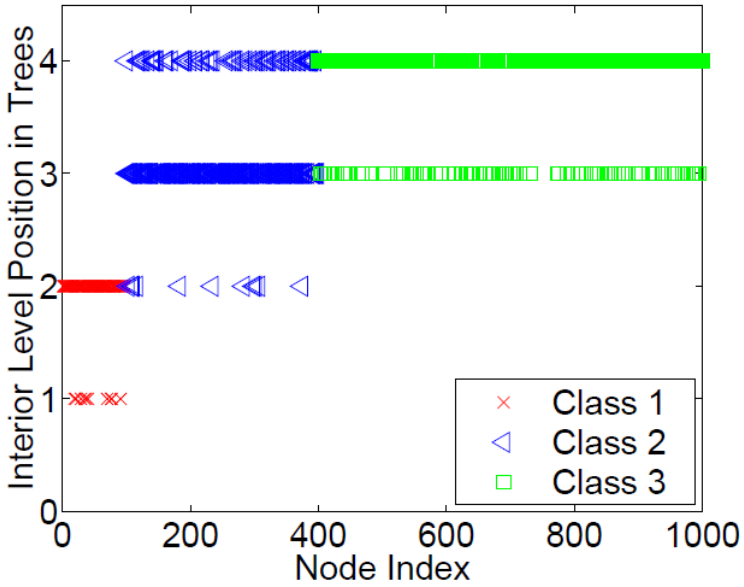


圖 10 最佳的節點放置方式

因此，將具有較大上載頻寬的節點放置在距離樹根較近的位置，確實能夠有效降低系統平均播放延遲時間。然而，這些研究不是假設網路中的節點具有相同的上載頻寬就是具有相同的點對點延遲時間，這樣的假設與真實網際網路的情況並不相符。

若欲在樹狀網路中提供 LS 服務，又想要保證使用者的播放延遲時間都能低於一個給定的延遲上限值，勢必會限制系統的容量。在接下來的章節中我們將介紹我們的研究工作。首先，我們提出了兩個演算法，在樹狀網路中，考慮不同的節點可能有不同的上載頻寬與點對點延遲時間，期望藉由降低系統平均播放延遲時間的方式來提升系統容量，並透過模擬實驗來驗證所提演算法的可行性。此外，我們實作了一個基於樹狀網路的 P2P LS 系統雛型，該系統雛型提供後續的研究一個真實的網路實測平台。

第四章、演算法設計

我們考慮 P2P LS 系統中有一個以串流伺服器為樹根的樹狀網路。串流伺服器和所有節點須定期的將其播放延遲時間及剩餘上載頻寬回報給一台追蹤伺服器(Tracking server)。Tracking server 就像是 BitTorrent 中的 Tracker 一般，扮演著輔助節點交換 Piece 的角色。他會提供新加入網路的節點一個父節點候選清單(Parent candidate list, PCL)，讓新加入的節點取得清單後能夠自行以 Ping 的方式探測自己與父節點候選人間的點對點延遲時間。新加入的節點須加總父節點候選人的播放延遲時間及彼此的播放延遲時間，並從所有父節點候選人中選擇一個父節點，使得自己有最小的播放延遲時間。此外，為了反映播放延遲時間的大小對於 LS 服務的重要性，我們設定一個延遲時間上限值(Maximum delay constraint)，並要求播放延遲時間小於延遲時間上限值的節點才能加入網路。如圖 11 所示，不同的節點擁有不同的上載頻寬和點對點延遲時間，且延遲時間上限值被設定為 300 毫秒。黑色的節點為播放延遲時間小於延遲時間上限值的節點，而紅色的節點則是播放延遲時間大於延遲時間上限值的節點，系統並不提供紅色的節點即時影音串流的服務。

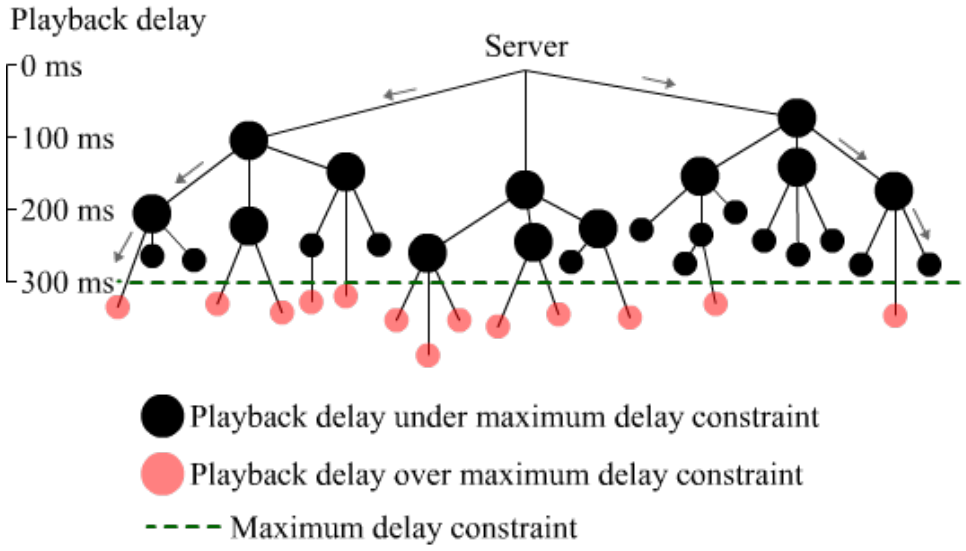


圖 11 延遲時間上限值

4.1 問題模型

P2P 網路中的成員包含一台串流伺服器 p_0 及一群節點 $p_1, p_2, \ldots, p_N$。不同的成員有不同的上載頻寬 b_i，i 分別為 0 到 N。由於成員須貢獻其上載頻寬，我們定義成員的剩餘上載頻寬為 r_i，i 分別為 0 到 N。此外，不同成員間有不同的點對點延遲時間，我們定義 p_i 和 p_j 間的點對點延遲時間為 $d_{i,j}$，而 p_i 的播放延遲時間被定義為 d_i。若提供資料率為 R 的即時影音串流，本章節的目的為找到一個以串流伺服器 p_0 為樹根的 Degree-constraint spanning tree 來連接節點 $p_1, p_2, \ldots, p_N$，使得系統具有最小的平均播放延遲時間，即最小的 $\sum_{i=0}^{N} d_i / N$。

將具有不同上載頻寬的節點用一個 Degree-constraint spanning tree 連接起來是一個 NP-complete 的問題[30]。因此，要將樹狀網路最佳化使其擁有最小平均播放延遲時間是不容易的。此外，在 P2P 應用中，節點可能在任何時間點加入或離開網路，所以試圖讓樹狀網路永遠保持最佳化是不可能的。更何況，實際要將樹狀網路最佳化必須先擁有任意兩個節點間的點對點延遲時間，這在規模較小的服務上尚有實現的可能，但在大規模的應用上是不可能實現的。

4.2 父節點候選清單生成演算法

透過節點的資訊回報機制，Tracking server 可以取得部分的樹狀網路資訊，包括所有節點的播放延遲時間及剩餘上載頻寬。根據這些有限的資訊，我們提出了一個 PCL 生成演算法，讓新加入網路的節點能夠被分配到適當的樹狀位置。

為了產生一個長度為 L 的 PCL，Tracking server 先取得新加入節點 *P* 的上載頻寬，並與目前網路中的其他節點相比較，令網路中有 *K* 個節點的上載頻寬大於節點 *P*。接著，Tracking server 將尚有剩餘頻寬的節點以播放延遲時間由小至大排序，令排序結果為 *BwEnough*。PCL 是 *BwEnough* 的子清單，包含了 *BwEnough* 中的第 Max(*K*-*L*/2, 1)至第 Min(*K*+*L*/2, |*BwEnough*|)個元素。演算法 1 中完整詳述了父節點候選清單生成演算法的內容。

演算法 1 父節點候選清單生成演算法

$P \leftarrow$ The newly joining peer

$L \leftarrow$ PCL Length

$R \leftarrow$ Streaming rate

$K \leftarrow 0$

$BwEnough \leftarrow \emptyset$

for $i = 1 \rightarrow N$ **do**

  if $b_i \geqq b_P$ **then** $K \leftarrow K+1$

  if $r_i \geqq R$ **then** $BwEnough \leftarrow BwEnough \cup \{p_i\}$

end for

Sort *BwEnough* in increasing order by playback delay

$PCL \leftarrow$ SubList(*BwEnough*, Max(*K*-*L*/2, 1), Min(*K*+*L*/2, |*BwEnough*|))

這個演算法的主要構想除了期望降低系統平均播放延遲時間外，也期望提供一個注重公平性的 LS 服務：能貢獻較多上載頻寬的節點應該有較低的播放延遲時間。因此，能貢獻較多上載頻寬的節點在加入網路時，從 Tracking server 取得的 PCL 中將包含許多播放延遲時間較小的父節點候選人。這個演算法鼓勵使用者踴躍貢獻可用的上載頻寬，而不是使用一些軟體來限制上載的網路流量。

4.3 區域最佳化演算法

除了透過 Tracking server 提供 PCL 來達到適當配置節點位置的目的外，我們也期望透過分散式的區域最佳化演算法進一步降低系統的平均播放延遲時間。我們令加入網路後的節點，每隔一段時間須執行一次區域最佳化。如圖 12 所示，對於正在執行區域最佳化的節點 P，找到一個以 P 之父節點 G 為樹根的 Degree-constraint spanning tree 來連接 P 及 P 的子節點 $C = \{C_1, C_2, \ldots, C_k\}$，使得該區域具有最小的平均播放延遲時間。

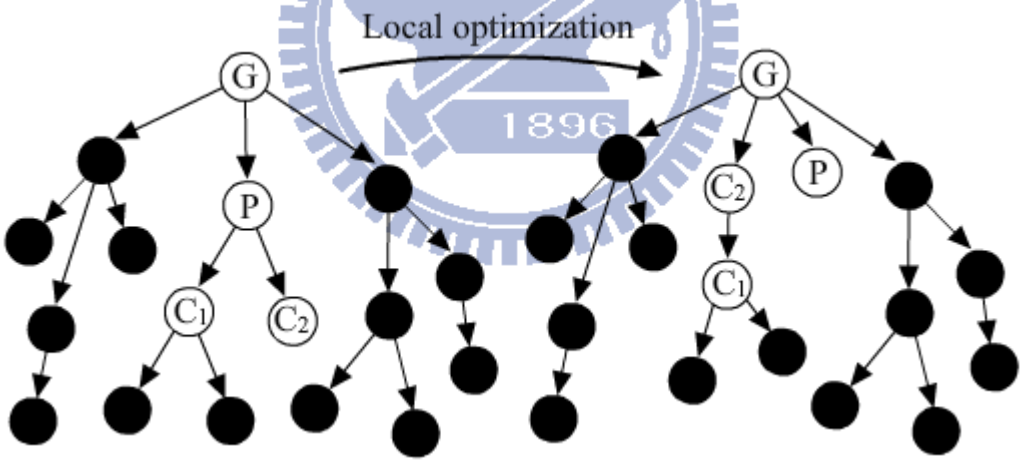


圖 12 區域最佳化

我們共提出了四種可能的演算法以供比較，並從中選擇較合適者成為區域最佳化演算法。在演算法 2 中，我們提供一個基於隨機排程的 Random 演算法。在演算法 3 中，我們提供一個考慮最大上載頻寬的 Largest upload bandwidth 演算法。在演算法 4 中，我們提供一個考慮最小播放延遲時間的 Lowest playback delay 演算法。在演算法 5 中，我們以列舉的方式求得區域最佳化的最佳解(Optimal)。

演算法 2 Random 演算法

AvailableParent ← {*G*}

Children ← {*P*} ∪ *C*

while *AvailableParent* ≠ *Ø* **and** *Children* ≠ *Ø* **do**

PCandidate ← Randomly pick a peer from *AvailableParent*

CCandidate ← Randomly pick a peer from *Children*

Set *PCandidate* to be the new parent of *CCandidate*

$d_{CCandidate} \leftarrow d_{PCandidate} + d_{PCandidate,CCandidate}$

$r_{PCandidate} \leftarrow r_{PCandidate} - R$

if $r_{PCandidate} < R$ **then** *AvailableParent* ← *AvailableParent* – {*PCandidate*}

Children ← *Children* – {*CCandidate*}

if $r_{CCandidate} \geqq R$ **then** *AvailableParent* ← *AvailableParent* ∪ {*CCandidate*}

end while

if *Children* = *Ø* **and** new local avg playback delay is lower than the origin **then**

Commit this optimization

end if

演算法 3 Largest upload bandwidth 演算法

AvailableParent ← {*G*}

Children ← {*P*} ∪ *C*

while *AvailableParent* ≠ *Ø* **and** *Children* ≠ *Ø* **do**

CCandidate ← *Peer* ∈ *Children* where b_{Peer} is maximum

PCandidate ← *Peer* ∈ *AvailableParent* where $d_{Peer} + d_{Peer,CCandidate}$ is minimum

Set *PCandidate* to be the new parent of *CCandidate*

$d_{CCandidate} \leftarrow d_{PCandidate} + d_{PCandidate,CCandidate}$

$r_{PCandidate} \leftarrow r_{PCandidate} - R$

if $r_{PCandidate} < R$ **then** *AvailableParent* ← *AvailableParent* – {*PCandidate*}

Children ← *Children* – {*CCandidate*}

if $r_{CCandidate} \geqq R$ **then** *AvailableParent* ← *AvailableParent* ∪ {*CCandidate*}

end while

if *Children* = *Ø* **and** new local avg playback delay is lower than the origin **then**

Commit this optimization

end if

演算法 4 Lowest playback delay 演算法

AvailableParent ← {*G*}

Children ← {*P*} ∪ *C*

while *AvailableParent* ≠ *Ø* **and** *Children* ≠ *Ø* **do**

PCand ∈ *AvailableParent*, *CCand* ∈ *Children*

Find the best (*PCand*, *CCand*) pair such that $d_{PCand} + d_{PCand,CCand}$ is minimum

Set *PCand* to be the new parent of *CCand*

$d_{CCand} \leftarrow d_{PCand} + d_{PCand,CCand}$

$r_{PCand} \leftarrow r_{PCand} - R$

if $r_{PCand} < R$ **then** *AvailableParent* ← *AvailableParent* – {*PCand*}

Children ← *Children* – {*CCand*}

if $r_{CCand} \geqq R$ **then** *AvailableParent* ← *AvailableParent* ∪ {*CCand*}

end while

if *Children* = *Ø* **and** new local avg playback delay is lower than the origin **then**

Commit this optimization

end if

演算法 5 Optimal 演算法

AvailableParent ← {*G*}

Children ← {*P*}∪*C*

Use brute force method to find the optimal spanning tree

Commit the optimization

我們以模擬實驗的方式來比較上述四種區域最佳化演算法。在模擬環境中，我們假設每次最佳化中，包含 G、P 和 P 的所有子節點，且 P 平均有 1 到 4 個子節點，他們之間的點對點延遲時間均勻分佈在 1 毫秒至 1000 毫秒之間。由於所提供的串流資料率(Data rate)不同，不同演算法可能有不同的表現結果，因此我們分別考慮三種不同的情境。

從第一種至第三種情境中所提供的 Data rate 越來越低，也就是每個節點平均可服務的子節點個數越來越多。在第一種情境中，每個節點可服務的子節點個數平均分佈在 0 個到 2 個之間。在第二種情境中，每個節點可服務的子節點個數平均分佈在 0 個到 4 個之間。在第三種情境中，每個節點可服務的子節點個數平均分佈在 0 個到 8 個之間

如圖 13 所示，Random 演算法在四種情境中的平均播放延遲時間都是最高的。Largest upload bandwidth 演算法在四種情境中有著次高的平均播放延遲時間。Lowest playback delay 演算法在四種情境中有第二小的平均播放延遲時間。而 Optimal 演算法有著最低的平均播放延遲時間。

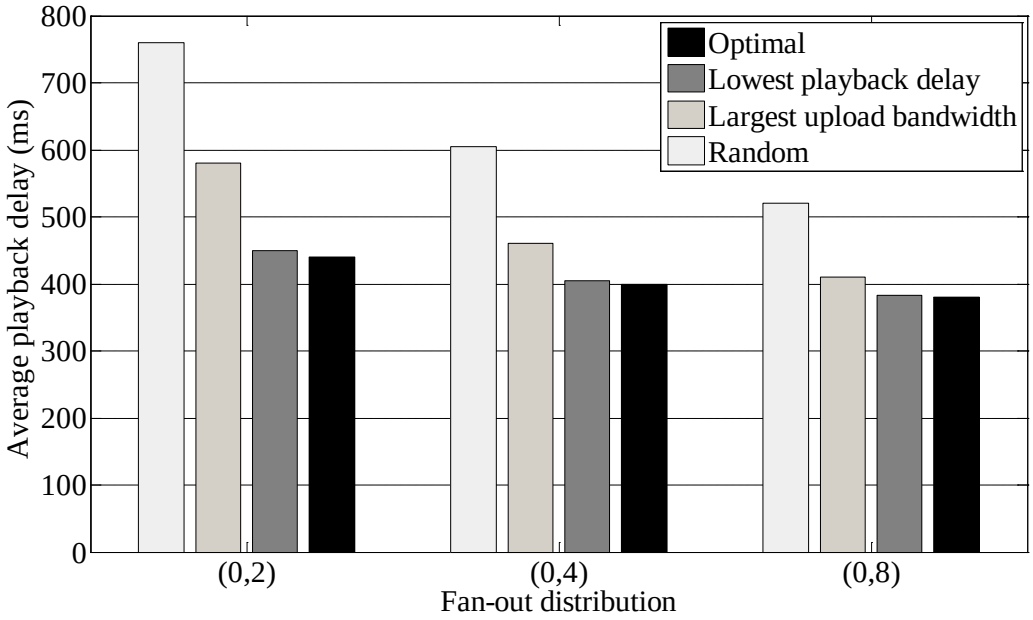


圖 13 區域最佳化演算法比較

Lowest playback delay 演算法雖然表現不如列舉，但其平均播放延遲時間僅高於 Optimal 演算法兩個百分比，而計算複雜度遠小於 Optimal 演算法。因此，我們採用 Lowest playback delay 演算法作為區域最佳化演算法。

第五章、實驗模擬

在本章節中，我們將透過模擬實驗的方式模擬 PCL 生成演算法與區域最佳化演算法相整合的組合效果。在上個章節末段，我們發現使用 Lowest playback delay 演算法來達到區域最佳化，對於降低系統平均播放延遲時間的效果趨近於最佳解。我們考慮若能以相同的概念，在新節點加入網路時就將節點配置在適當的位置，是否也能夠有效降低系統平均播放延遲時間。因此，我們設計一個用來與上個章節所提兩個演算法的組合相比較的 Greedy 演算法。在 Greedy 演算法中，我們假設 Tracking server 知道網路中任兩個節點間的點對點延遲時間。一旦節點加入網路，Tracking server 會告知該節點只須與誰相連接就能擁有最小的播放延遲時間，詳細演算法內容如演算法 6 所示。

演算法 6 Greedy 演算法

$P \leftarrow$ The newly joining peer

$R \leftarrow$ Streaming rate

$BwEnough \leftarrow \emptyset$

for $i = 1 \rightarrow N$ **do**

  if $r_i \geqq R$ **then** $BwEnough \leftarrow BwEnough \cup \{p_i\}$

end for

Find the best $Parent \in BwEnough$ such that $d_{Parent} + d_{Parent,P}$ is minimum

5.1 實驗環境設定

我們參考[31]來設定節點間的點對點延遲時間分佈，如表 1 所示，28%的連線有 10 毫秒的延遲時間、60%的連線有 50 毫秒的延遲時間，剩餘 12%的連線有 100 毫秒的延遲時間。此外，我們參考中華電信 2010 年營運報告[32]設定節點的上載頻寬分佈，30%的節點上載頻寬為 384Kbps、10%的節點上載頻寬為 640Kbps 及 60%的節點上載頻寬為 2Mbps，如表 2 所示。

表 1 點對點延遲時間分佈機率

End-to-End delay	10 毫秒	50 毫秒	100 毫秒
百分比	28	60	12

表 2 上載頻寬分佈機率

上載頻寬	384Kbps	640Kbps	2Mbps
百分比	30	10	60

5.2 實驗結果

為了模擬 P2P 網路的動態環境，我們使用 Poisson 分佈，使得每秒有 4 個節點加入網路。並使用 Exponential 分佈，讓網路中的節點停留約 600 秒後才離開網路。根據 Little formula，在確保上載頻寬充足且未設定延遲時間上限值的情況下，穩定後的樹狀網路中約有 2400 個節點，因此我們假定 2400 為最佳的系統容量。如表 4 所示，在不同 Data rate 及不同延遲時間上限值時，我們以模擬實驗比較使用清單長度為 20 的 PCL 生成演算法(記為 PCL)、含區域最佳化的 PCL 演算法(記為 OPT)及 Greedy 演算法(記為 Greedy)在四種不同情境下的效果如何。

表 3 四種網路模擬情境

	未設定延遲時間上限值	延遲時間上限值為 200 毫秒
500Kbps data rate	情境一	情境二
1Mbps data rate	情境三	情境四

5.3.1 情境一、500Kbps 串流資料率及未設定延遲時間上限值

在 Data rate 為 500Kbps 的情況下，共有 70%的節點能夠成為樹狀網路中的內部節點，唯有 30%的節點僅能作為葉子節點。其中上載頻寬為 640Kbps 的節點可以再服務 1 個子節點，而上載頻寬為 2Mbps 的節點可以再服務 4 個子節點，因此系統中平均每個線上的節點可以再服務 2.5 個子節點，可謂是不乏上載頻寬。此外，此情境中並未設定延遲時間上限值，欲加入網路的節點只要能夠找到尚有剩餘上載頻寬的父節點就可以成功加入網路。

如圖 14 所示，不管是使用 Greedy 演算法、PCL 演算法或是 OPT 演算法，系統都能容納約 2400 個節點。如圖 15 所示，PCL 演算法及 OPT 演算法的系統平均播放延遲時間分別比 Greedy 演算法低 8.3%及 16.6%。如圖 16 所示，Greedy 演算法並未提供一個注重公平性的 LS 服務，不管節點的上傳頻寬多寡都有相同的平均播放延遲時間。如圖 17 所示，若採用 PCL 演算法，具有 2Mbps 上載頻寬的節點能貢獻較多上載頻寬，因此其平均播放延遲時間僅有 40 毫秒，低於系統平均的 55 毫秒。而具有 384Kbps 上載頻寬的節點未能貢獻任何上載頻寬，因此其平均播放延遲時間為 80 毫秒，高於系統平均。如圖 18 所示，若採用 OPT 演算法，具有 2Mbps 上載頻寬的節點能貢獻較多上載頻寬，因此其平均播放延遲時間僅有 38 毫秒，低於系統平均的 50 毫秒。而具有 384Kbps 上載頻寬的節點未能貢獻任何上載頻寬，因此其平均播放延遲時間為 69 毫秒，高於系統平均。

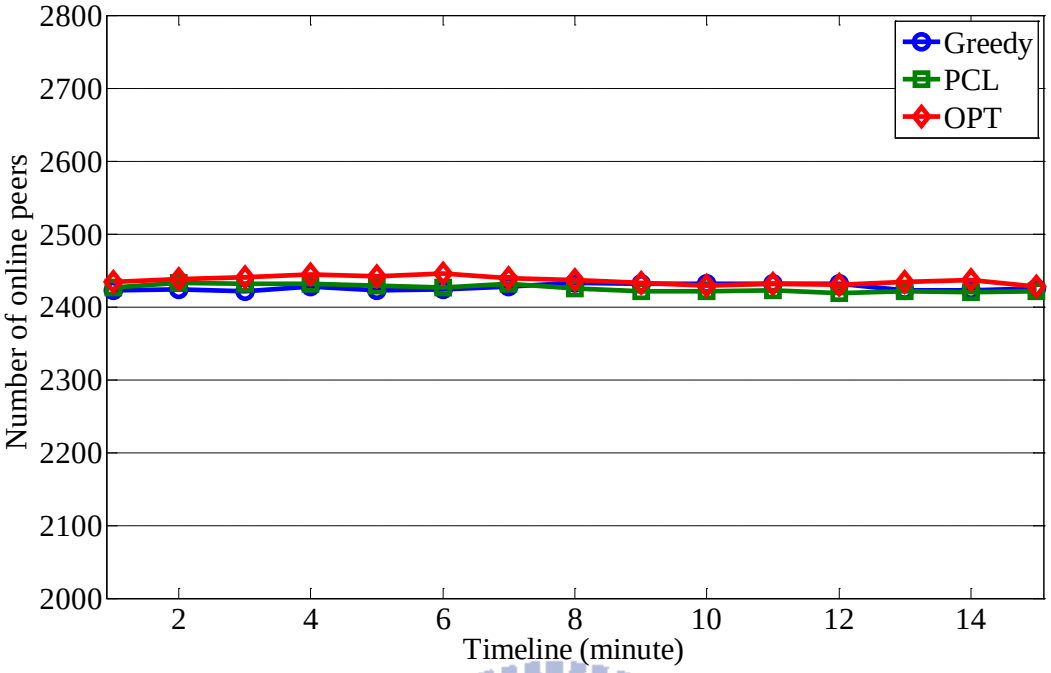


圖 14 情境一之線上節點個數

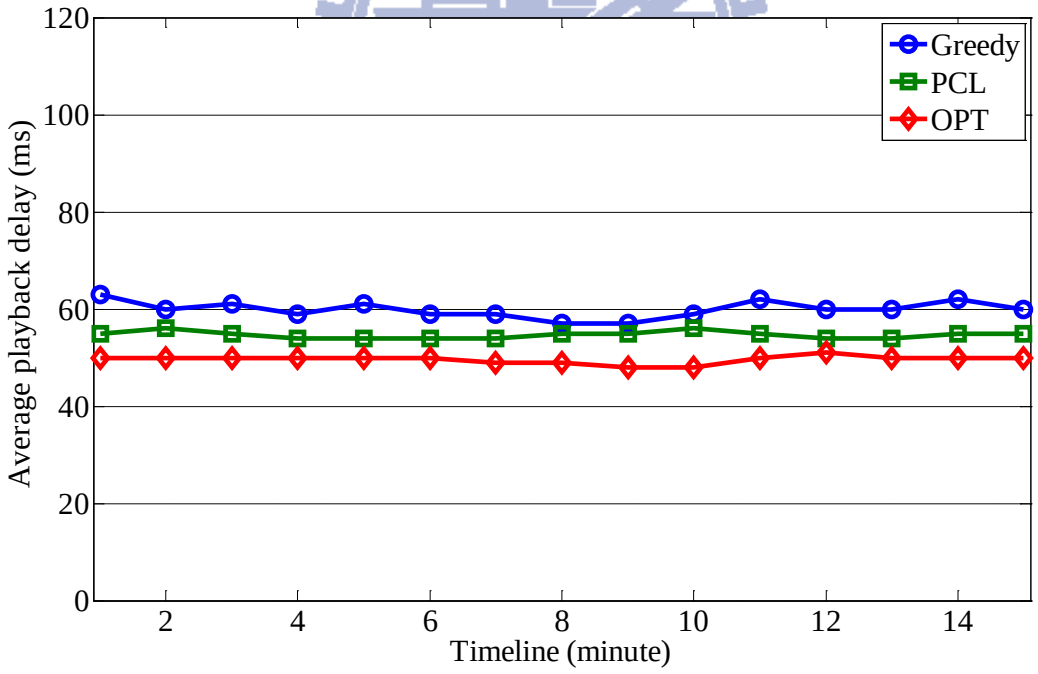


圖 15 情境一之系統平均 Playback delay

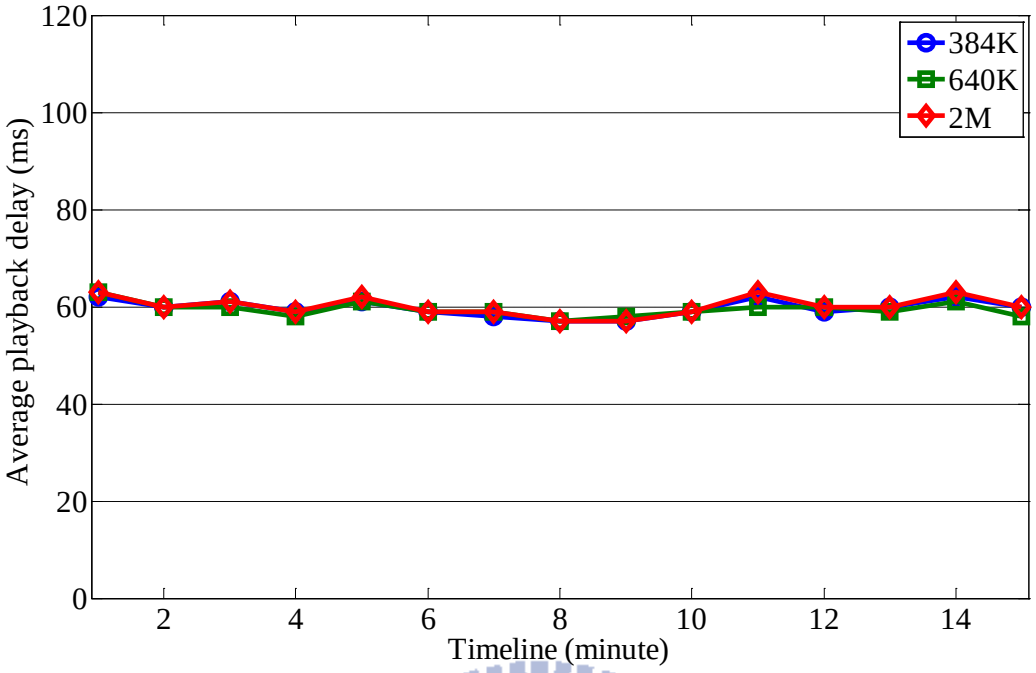


圖 16 情境一中 Greedy 的平均 Playback delay

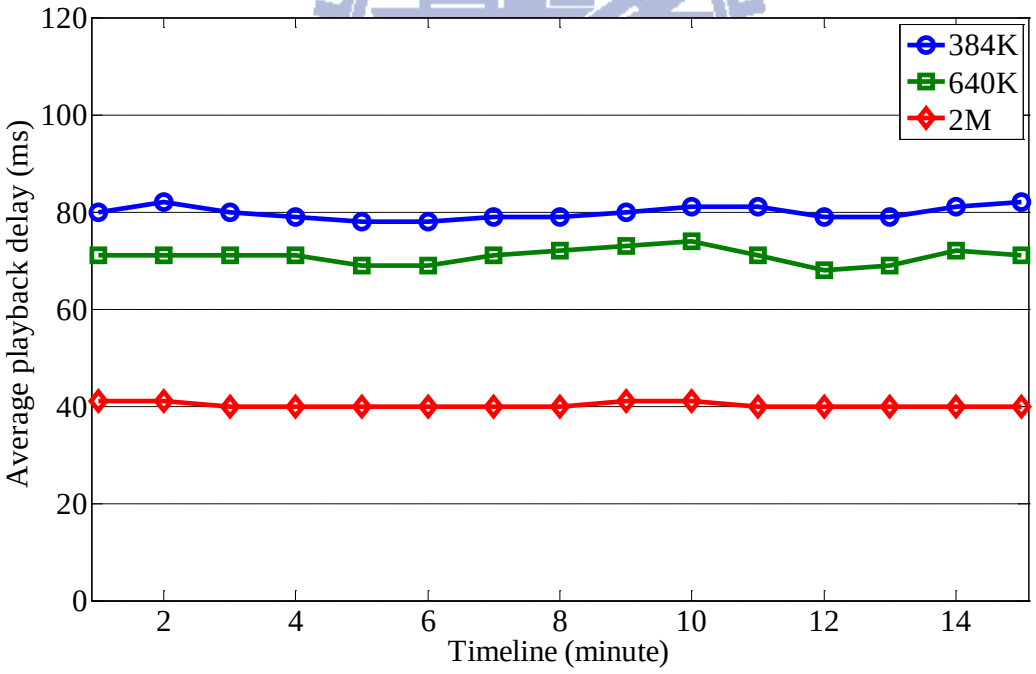


圖 17 情境一中 PCL 的平均 Playback delay

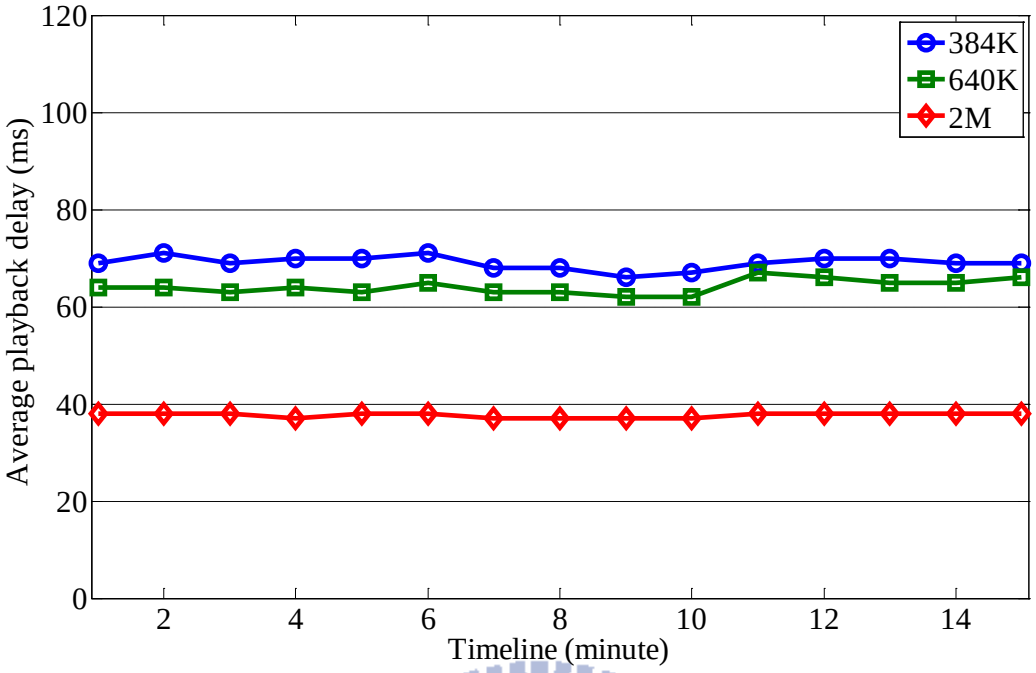


圖 18 情境一中 OPT 的平均 Playback delay

5.3.2 情境二、500Kbps 串流資料率及 200 毫秒延遲時間上限值

在 Data rate 為 500Kbps 的情況下，共有 70%的節點能夠成為樹狀網路中的內部節點，唯有 30%的節點僅能作為葉子節點。其中上載頻寬為 640Kbps 的節點可以再服務 1 個子節點，而上載頻寬為 2Mbps 的節點可以再服務 4 個子節點，因此系統中平均每個線上的節點可以再服務 2.5 個子節點，可謂是不乏上載頻寬。此外，此情境中設定延遲時間上限值為 200 毫秒，欲加入網路的節點除了須找到尚有剩餘上載頻寬的父節點外尚須保證其播放延遲時間低於 200 毫秒才能成功加入網路。

如圖 19 所示，不管是使用 Greedy 演算法、PCL 演算法或是 OPT 演算法，系統都能容納約 2400 個節點。如圖 20 所示，PCL 演算法及 OPT 演算法的系統平均播放延遲時間分別比 Greedy 演算法低 15.6%及 20.3%。如圖 21 所示，Greedy 演算法並未提供一個注重公平性的 LS 服務，不管節點的上傳頻寬多寡都有相同的平均播放延遲時間。如圖 22 所示，若採用 PCL 演算法，具有 2Mbps 上載頻寬的節點能貢獻較多上載頻寬，因此其平均播放延遲時間僅有 40 毫秒，低於系統平均的 54 毫秒。而具有 384Kbps 上載頻寬的節點未能貢獻任何上載頻寬，因此其平均播放延遲時間為 78 毫秒，高於系統平均。如圖 23 所示，若採用 OPT 演算法，具有 2Mbps 上載頻寬的節點能貢獻較多上載頻寬，因此其平均播放延遲時間僅有 39 毫秒，低於系統平均的 51 毫秒。而具有 384Kbps 上載頻寬的節點未能貢獻任何上載頻寬，因此其平均播放延遲時間為 72 毫秒，高於系統平均。

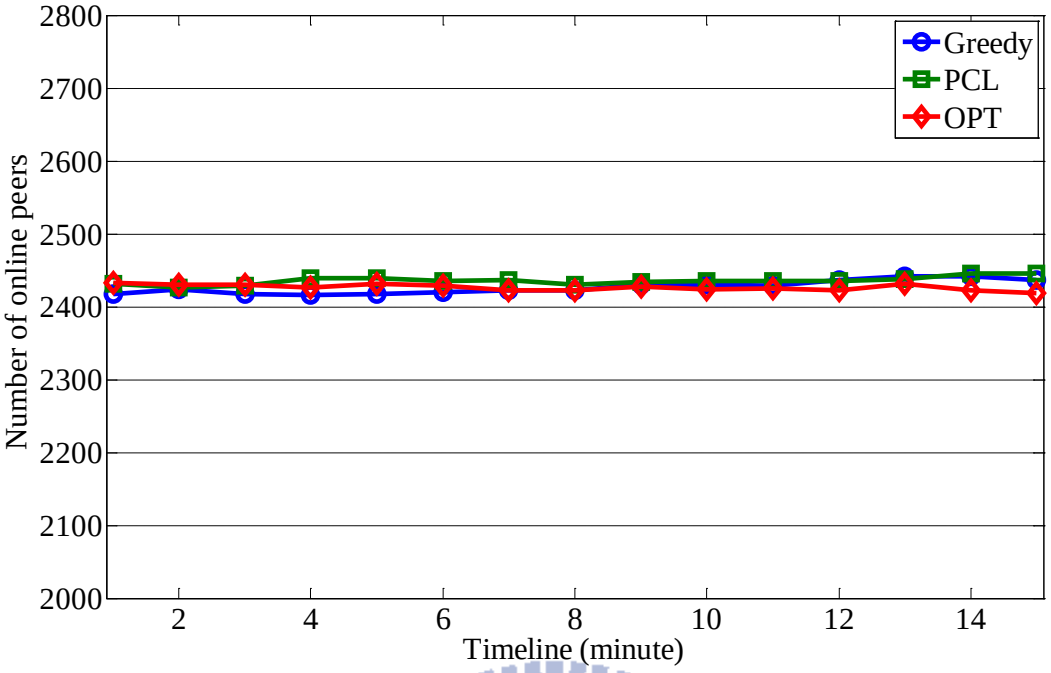


圖 19 情境二之線上節點個數

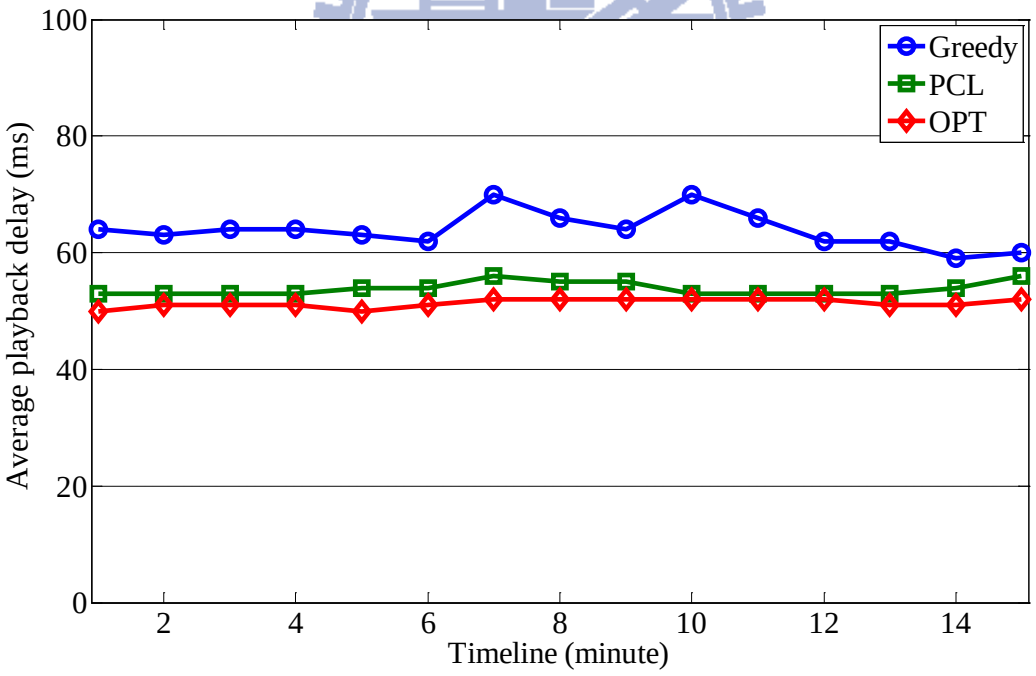


圖 20 情境二之系統平均 Playback delay

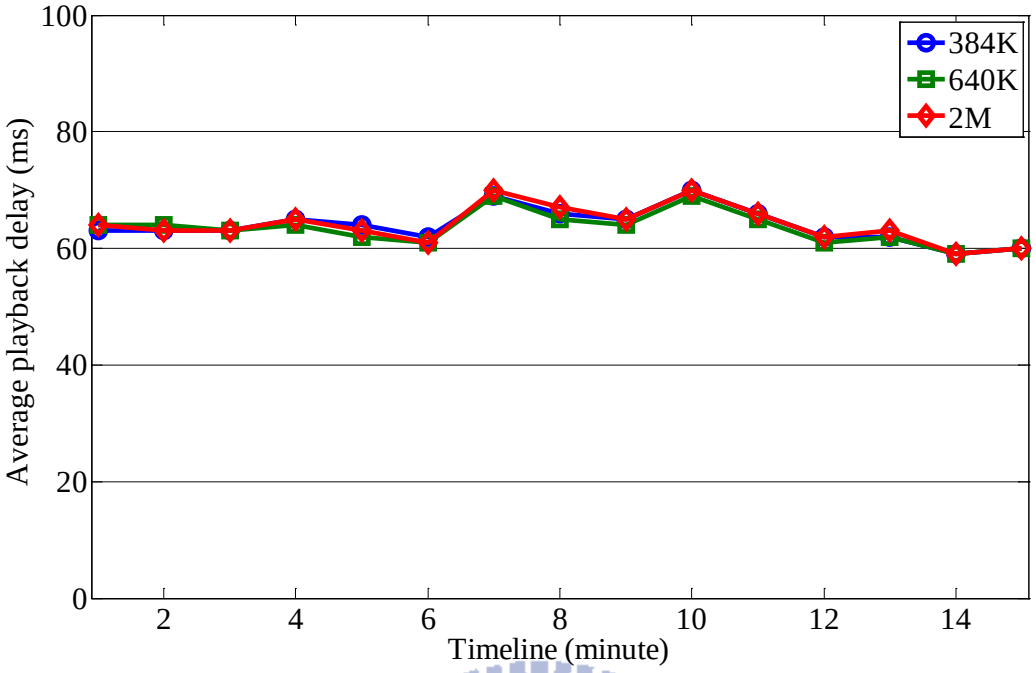


圖 21 情境二中 Greedy 的平均 Playback delay

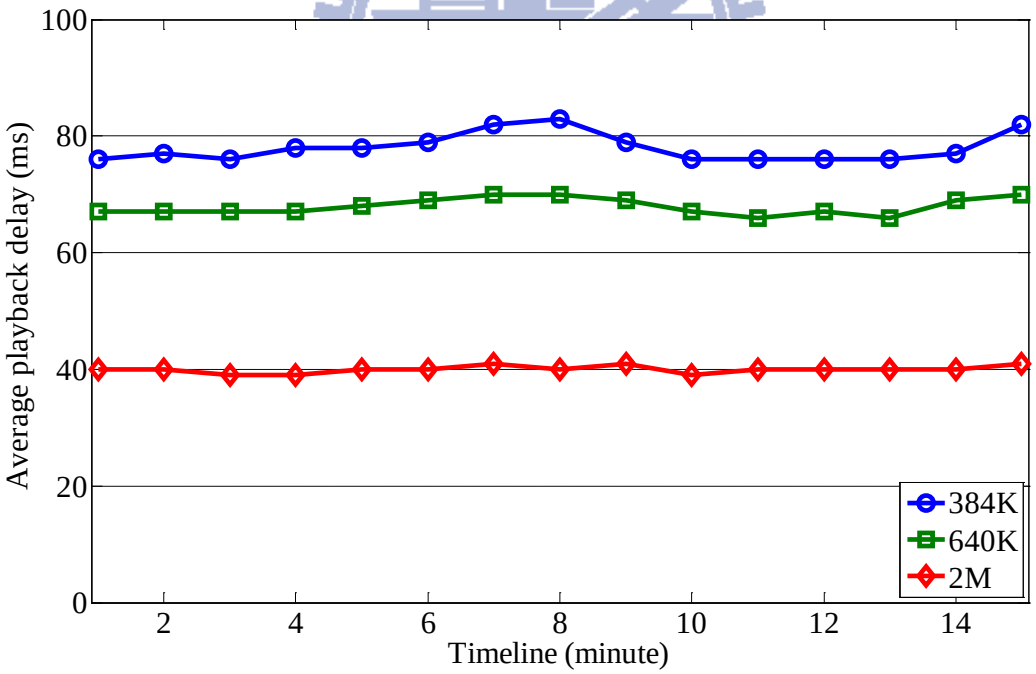


圖 22 情境二中 PCL 的平均 Playback delay

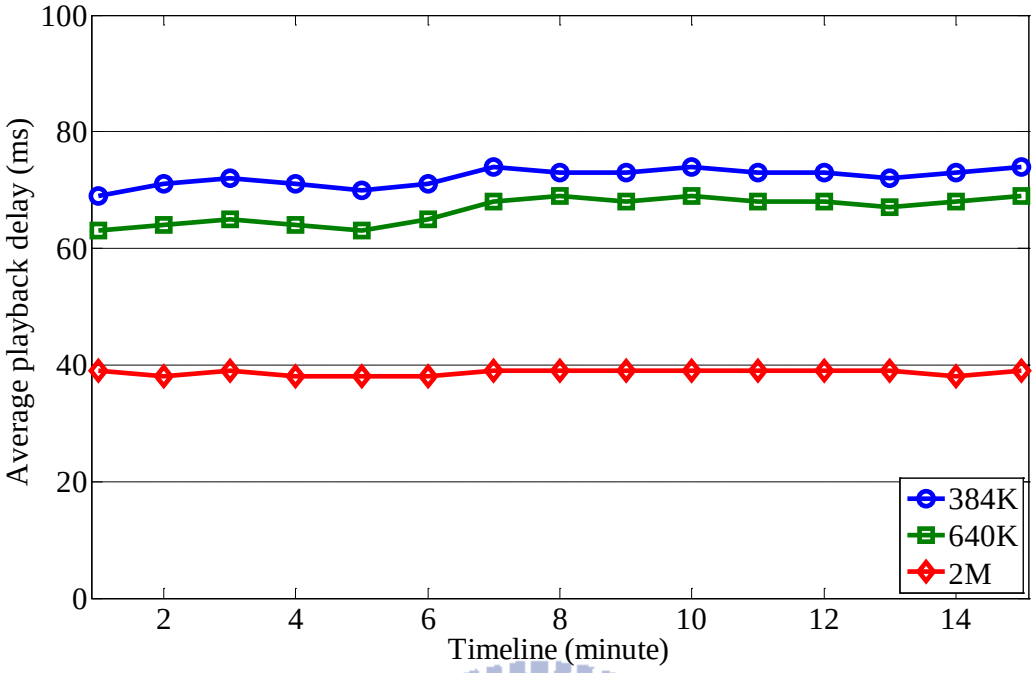


圖 23 情境二中 OPT 的平均 Playback delay

5.3.3 情境三、1Mbps 串流資料率及未設定延遲時間上限值

在 Data rate 為 1Mbps 的情況下，共有 60%的節點能夠成為樹狀網路中的內部節點，剩餘 40%的節點僅能作為葉子節點。其中僅有上載頻寬為 2Mbps 的節點可以再服務 2 個子節點，剩餘的節點都無法再服務任何子節點，因此系統中平均每個線上的節點可以再服務 1.2 個子節點，可謂是上載頻寬不足。此外，此情境中並未設定延遲時間上限值，欲加入網路的節點只要能夠找到尚有剩餘上載頻寬的父節點就可以成功加入網路。

如圖 24 所示，不管是使用 Greedy 演算法、PCL 演算法或是 OPT 演算法，系統都能容納約 2400 個節點。如圖 25 所示，PCL 演算法及 OPT 演算法的系統平均播放延遲時間分別比 Greedy 演算法低 51.4%及 57.8%。如圖 26 所示，Greedy 演算法並未提供一個注重公平性的 LS 服務，不管節點的上傳頻寬多寡都有相同的平均播放延遲時間。如圖 27 所示，若採用 PCL 演算法，僅具有 2Mbps 上載頻寬的節點能貢獻上載頻寬，因此其平均播放延遲時間僅有 114 毫秒，低於系統平均的 131 毫秒。而具有 384Kbps 及 640Kbps 上載頻寬的節點未能貢獻任何上載頻寬，因此其平均播放延遲時間為 159 毫秒，高於系統平均。如圖 28 所示，若採用 OPT 演算法，僅具有 2Mbps 上載頻寬的節點能貢獻上載頻寬，因此其平均播放延遲時間僅有 97 毫秒，低於系統平均的 114 毫秒。而具有 384Kbp 及 640Kbps 上載頻寬的節點未能貢獻任何上載頻寬，因此其平均播放延遲時間為 141 毫秒，高於系統平均。

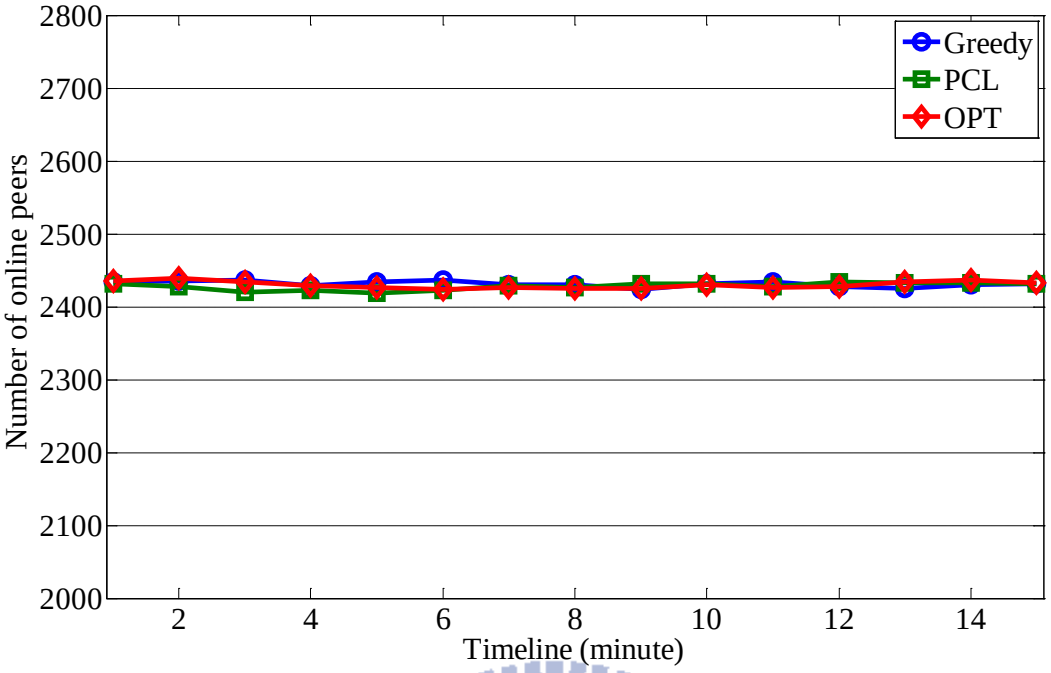


圖 24 情境三之線上節點個數

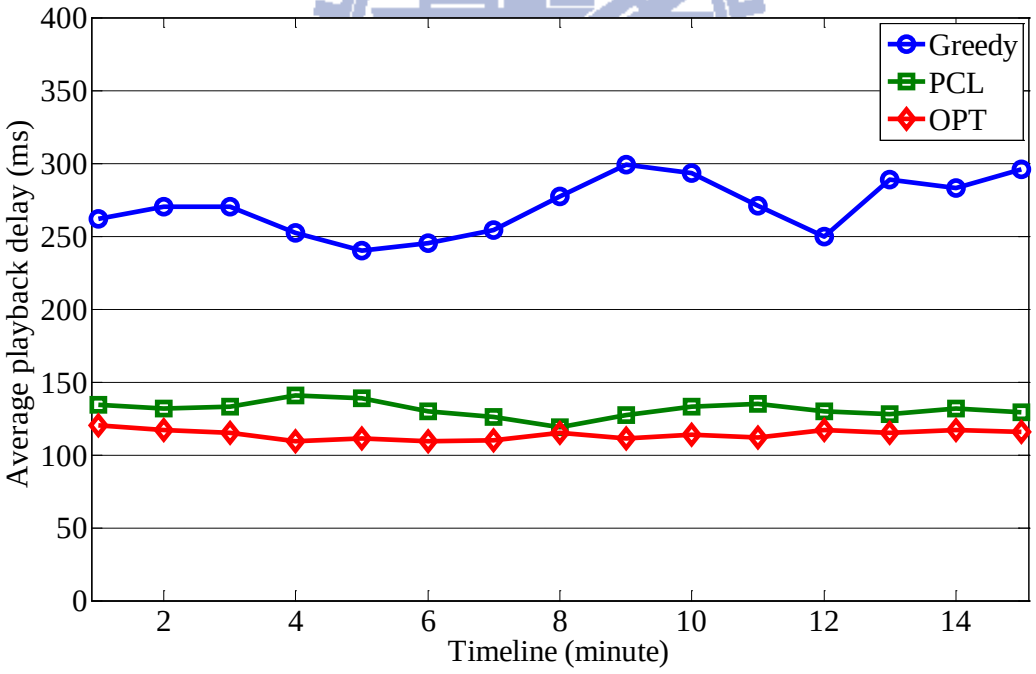


圖 25 情境三之系統平均 Playback delay

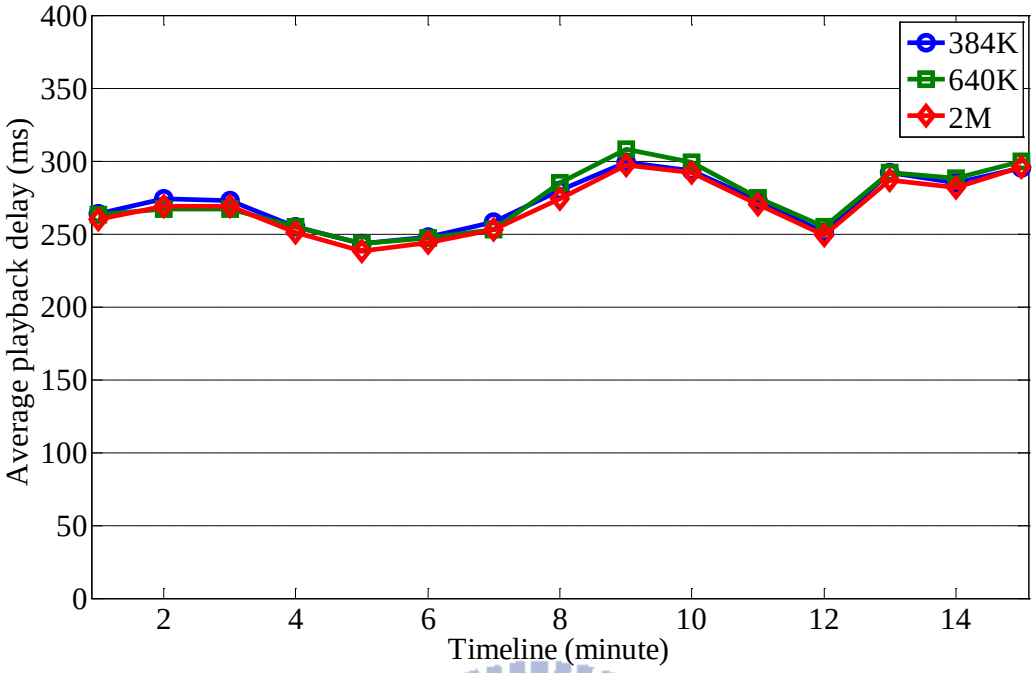


圖 26 情境三中 Greedy 的平均 Playback delay

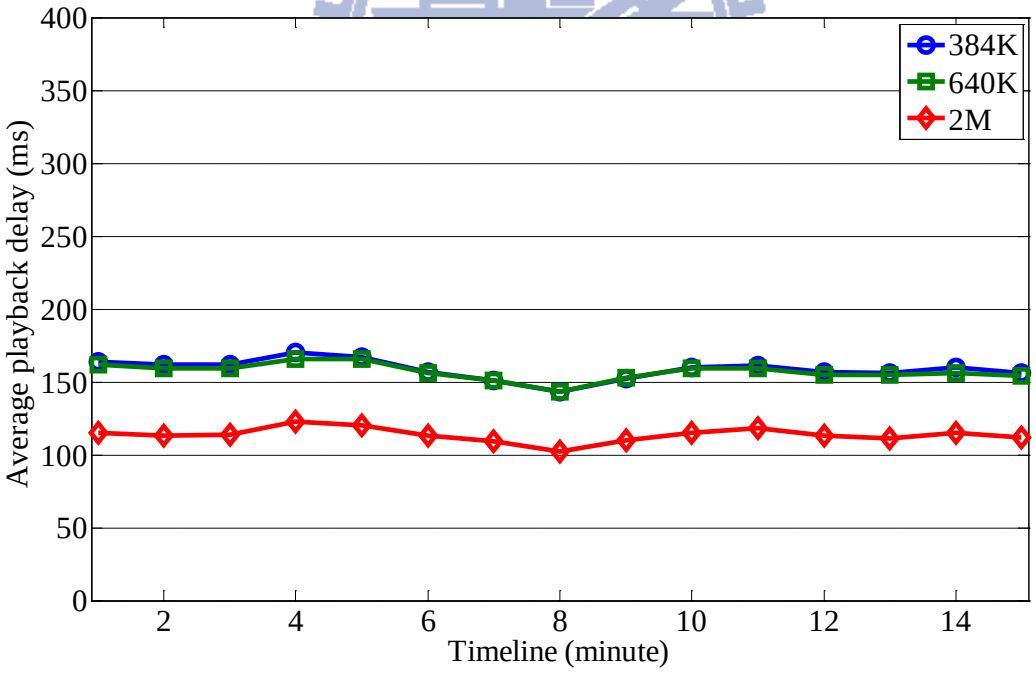


圖 27 情境三中 PCL 的平均 Playback delay

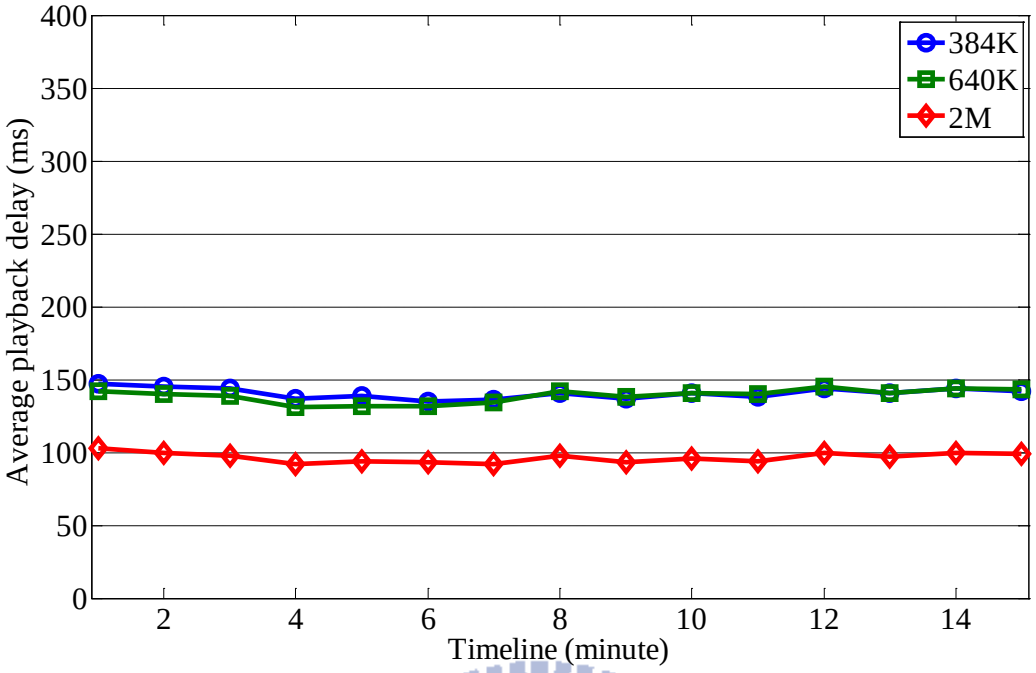


圖 28 情境三中 OPT 的平均 Playback delay

5.3.4 情境四、1Mbps 串流資料率及 200 毫秒延遲時間上限值

在 Data rate 為 1Mbps 的情況下，共有 60%的節點能夠成為樹狀網路中的內部節點，剩餘 40%的節點僅能作為葉子節點。其中僅有上載頻寬為 2Mbps 的節點可以再服務 2 個子節點，剩餘的節點都無法再服務任何子節點，因此系統中平均每個線上的節點可以再服務 1.2 個子節點，可謂是上載頻寬不足。此外，此情境中設定延遲時間上限值為 200 毫秒，欲加入網路的節點除了須找到尚有剩餘上載頻寬的父節點外尚須保證其播放延遲時間低於 200 毫秒才能成功加入網路。

如圖 29 所示，在採用 PCL 演算法及 OPT 演算法的系統都能容納約 2400 個節點，僅有採用 Greedy 演算法的系統僅能容納不到 500 個節點。如圖 30 所示，PCL 演算法及 OPT 演算法的系統平均播放延遲時間分別比 Greedy 演算法低 24%及 30%。如圖 31 所示，Greedy 演算法並未提供一個注重公平性的 LS 服務，不管節點的上傳頻寬多寡都有相同的平均播放延遲時間。如圖 32 所示，若採用 PCL 演算法，僅具有 2Mbps 上載頻寬的節點能貢獻上載頻寬，因此其平均播放延遲時間僅有 114 毫秒，低於系統平均的 127 毫秒。而具有 384Kbps 及 640Kbps 上載頻寬的節點未能貢獻任何上載頻寬，因此其平均播放延遲時間為 146 毫秒，高於系統平均。如圖 33 所示，若採用 OPT 演算法，僅具有 2Mbps 上載頻寬的節點能貢獻上載頻寬，因此其平均播放延遲時間僅有 105 毫秒，低於系統平均的 117 毫秒。而具有 384Kbp 及 640Kbps 上載頻寬的節點未能貢獻任何上載頻寬，因此其平均播放延遲時間為 136 毫秒，高於系統平均。

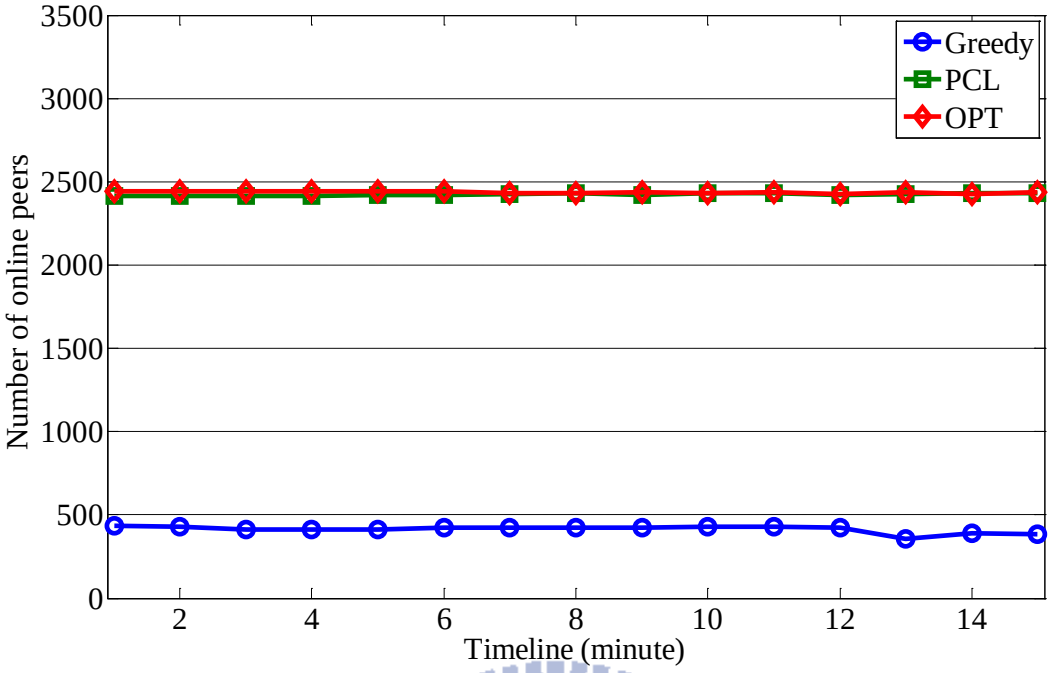


圖 29 情境四之線上節點個數

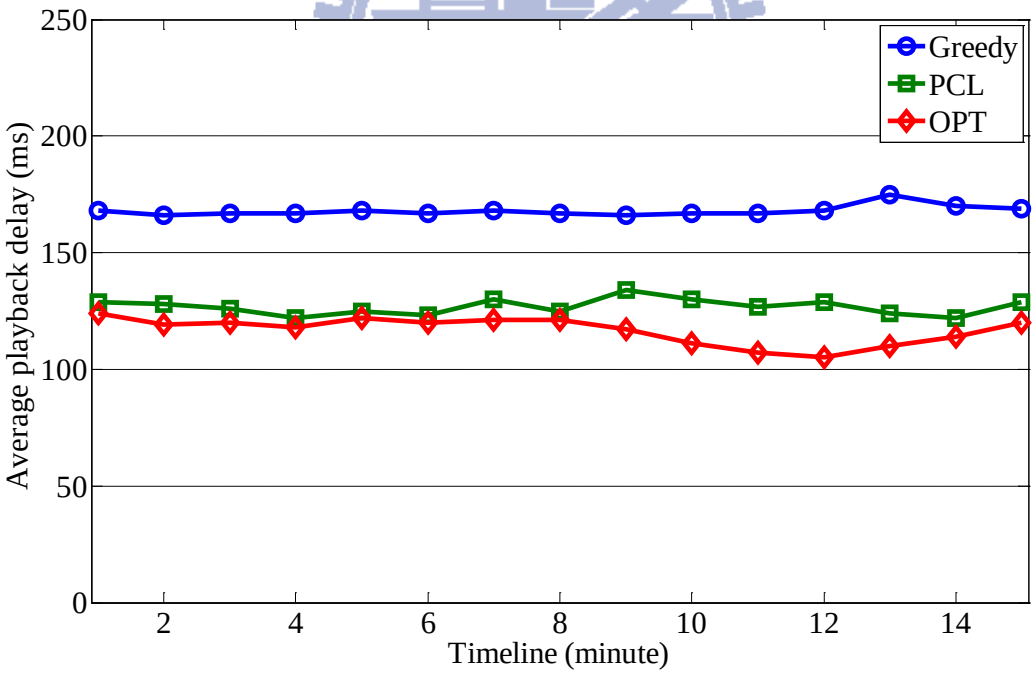


圖 30 情境四之系統平均 Playback delay

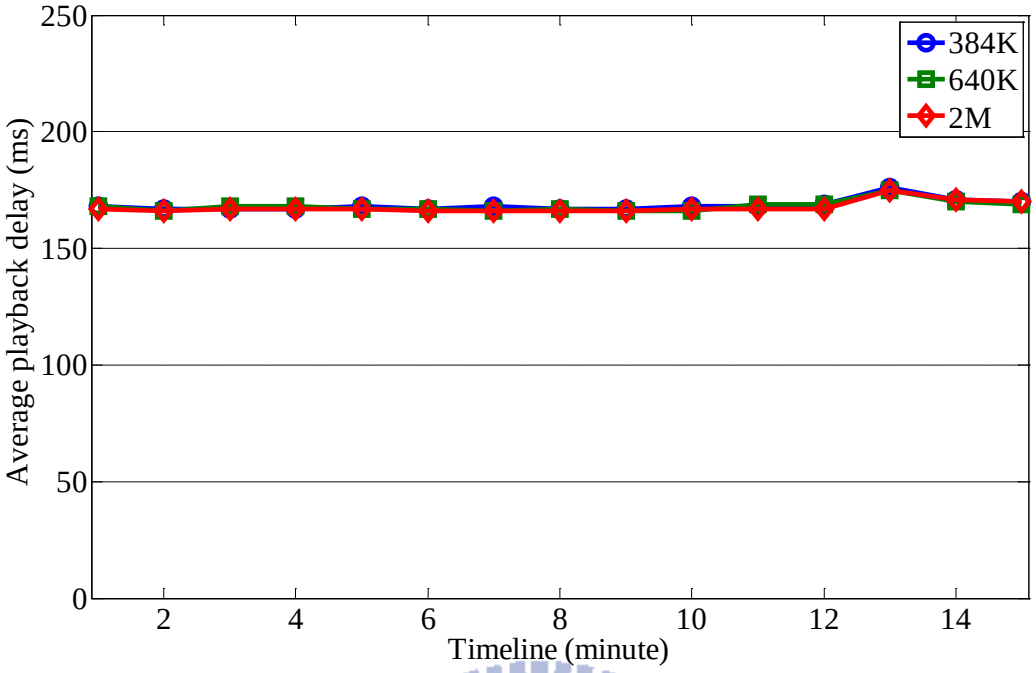


圖 31 情境四中 Greedy 的平均 Playback delay

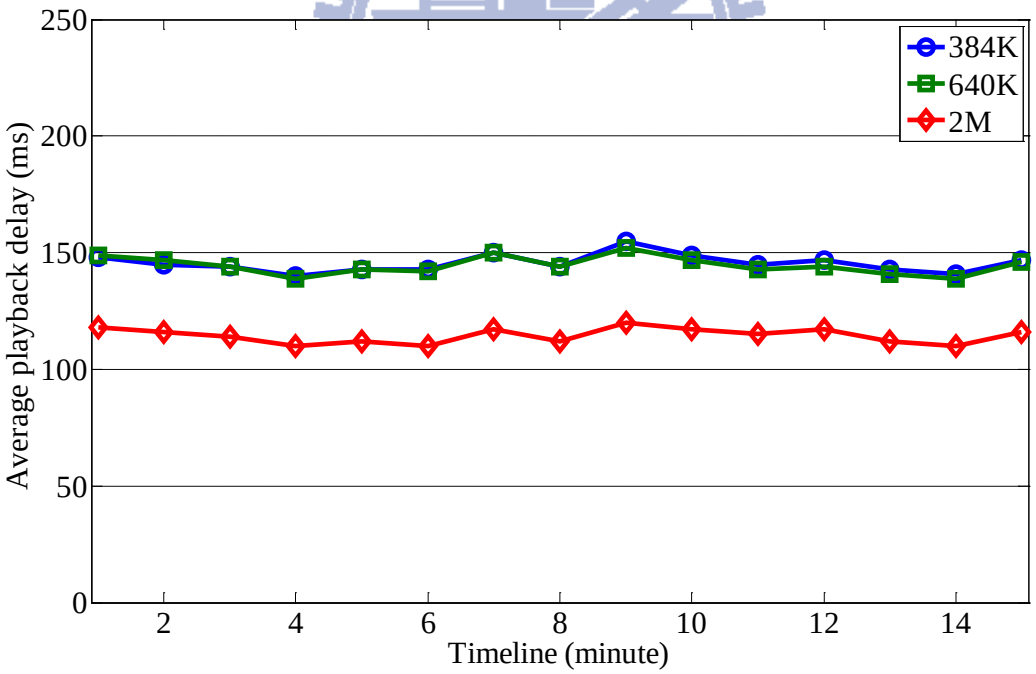


圖 32 情境四中 PCL 的平均 Playback delay

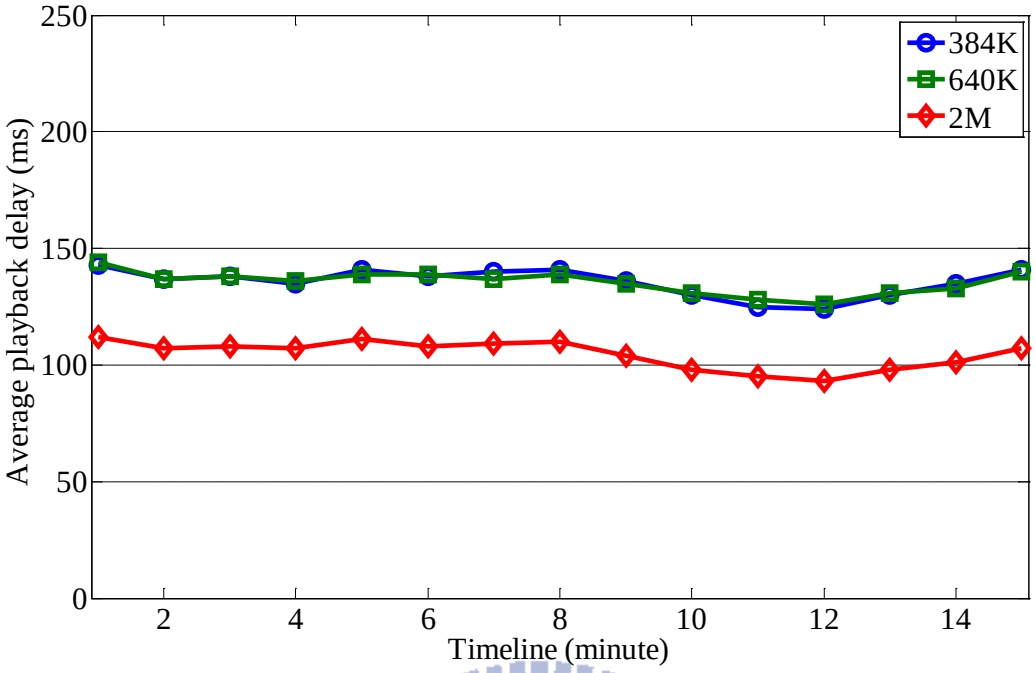


圖 33 情境四中 OPT 的平均 Playback delay

第六章、雛型系統設計

在 P2P 網路架構中，樹狀網路除了易於設計與實作外，也易於套用階層式的統計與管理機制。更重要的是，樹狀網路有助於計算節點的播放延遲時間。因此，我們設計了基於樹狀網路的 P2P LS 系統雛型。新設計 P2P 演算法可以被套用在雛型系統中，透過真實網路來評估演算法的好壞。或者，也可套用不同的串流緩衝與播放機制，透過真實網路來評量機制的好壞。本章節將介紹雛型系統的系統架構、軟體模組，並呈現雛型系統的實際運作畫面。

6.1 系統架構

雛型系統由三類元件所組成，分別是 Tracking server、Streaming server 及 Peer。在同個影音頻道(Channel)的節點會觀看到相同的即時影音串流，且頻道中的成員被一個樹狀網路連接起來。如圖 34 所示，網路中包含一個 Tracking server 及兩個影音頻道，分別是 Channel 1 及 Channel 2，並各自以一個樹狀網路來散佈即時影音串流。在樹狀網路中，樹根為 Streaming server，其他成員則為 Peer。影音串流由 Streaming server 透過樹狀拓撲散佈到網路中的所有 Peer。

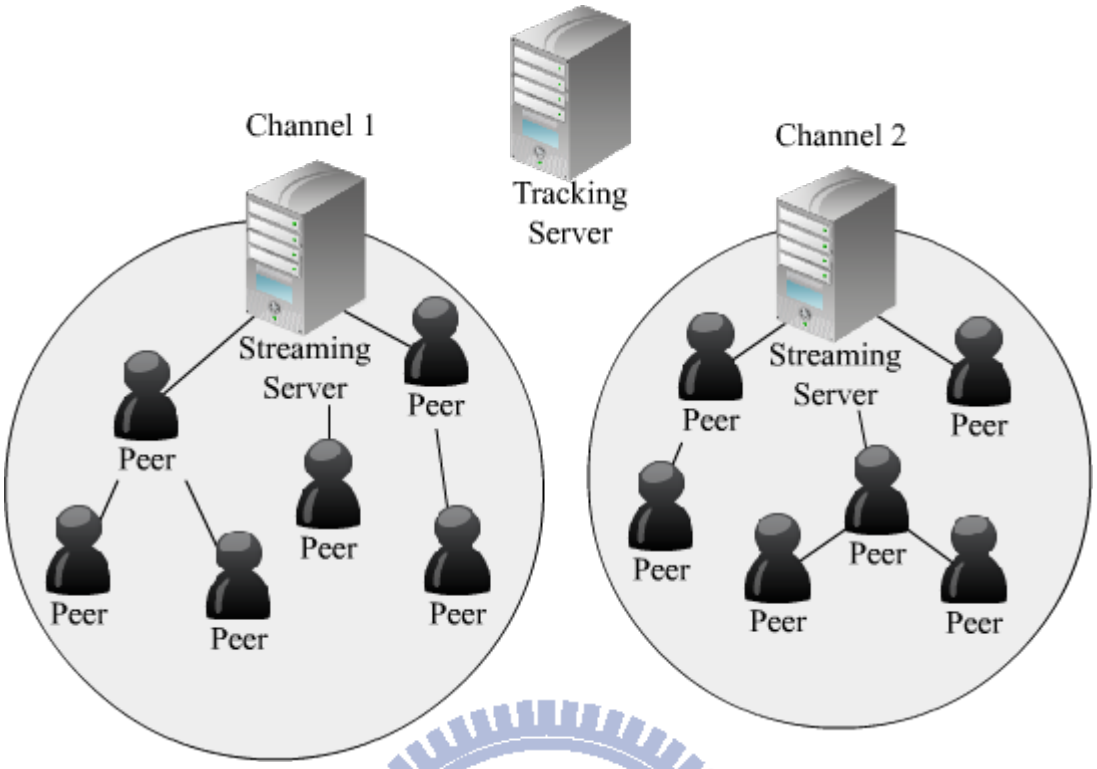


圖 34 雛型系統架構圖

6.1.1 Tracking server

一個 P2P 網路中須存在至少一個 Tracking server，其工作項目包括

1. 供 Streaming server 註冊影音頻道。
2. 提供 Peer 最新的影音頻道列表。
3. 提供 Peer 父節點候選清單。
4. 蒐集 Peer 的網路資訊，如播放延遲時間與剩餘上載頻寬等資訊。

6.1.2 Streamer server

Streaming server 是一個影音頻道的影音資料來源。一旦決定好影音資料的來源型態(如 File、Webcam 或 Camera on Cell Phone)，Streaming server 就可向 Tracking server 註冊一個新的影音頻道。完成影音頻道註冊後，Streaming server 則開始即時地產生影音資料，並將影音資料透過樹狀網路散佈給影音頻道中的所有節點。

6.1.3 Peer

Peer 在 P2P 網路中負責影音資料接收及轉送的工作。Peer 可自 Tracking server 取得目前的影音頻道列表，並從該列表中選擇一個影音頻道來觀看即時影音串流。決定欲加入的影音頻道後，Tracking server 會提供 Peer 一個 PCL，Peer 可從該清單中選擇其一作為其父節點，並從所選擇的父節點取得轉送的影音資料。同時，由於 Peer 未來亦可能擔任其他 Peer 的父節點，因此，Peer 也需具備轉送影音資料的能力。

6.2 軟體模組

本節將分別介紹 Tracking server、Streaming server 及 Peer 的軟體模組及各模組間如何協調及運作。

6.2.1 Tracker

Tracking server 是由 4 個模組所組成，分別是 Channel manager、Request dispatcher、Passive socket 及 Database。如圖 35 所示。

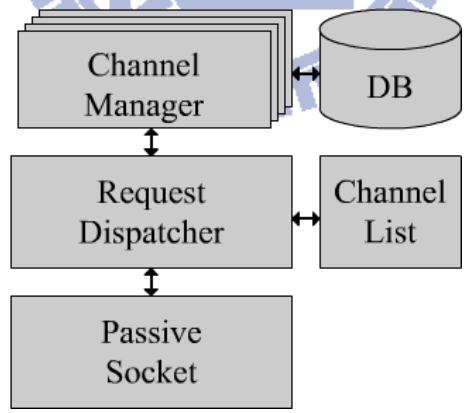


圖 35 Tracking server 的軟體模組圖

1. Passive socket

Tracking server 須和線上的 Streaming server 及 Peer 保持永久的連線，以便他們能夠回報其網路資訊。因此，Tracking server 須在某特定的通訊埠上開啟一個 passive socket，用來等待來自 Streaming server 及 Peer 建立連線的請求。

2. Request Dispatcher

Request dispatcher 負責將來自 Streaming server 與 Peer 的請求派送到相對應的 Channel manager，並將結果回傳。請求總類包含

1) Streaming server 請求註冊一個新的影音頻道；
2) Peer 請求提供最新的影音頻道清單；
3) 使用者的帳號登入請求；
4) Peer 請求加入或離開影音頻道；
5) Peer 更改父節點的通知；
6) 及 Peer 的 heart-beat 訊號，其中包含 Peer 的網路資訊，如播放延遲時間與剩餘上載頻寬等資訊。

3. Channel manager

Channel manager 負責管理所屬的影音頻道。如圖 36 所示，Peer map 包含影音頻道的網路拓撲及所有 Peer 的網路資訊。藉由 Topology monitor，Tracking server 的管理員可以檢視該影音頻道及時的網路拓撲。Peer 會定期將其網路資訊(如播放延遲時間、剩餘上載頻寬、目前的父節點)回報給 Tracking server，此時 Heart-beat handler 會將新的 Peer 資訊更新至 Peer map。每當有 Peer 欲加入該影音頻道，Peer churn handler 會要求 PCL generator 產生一個 PCL，並將此清單提供給 Peer。若有 Peer 欲離開頻道，Peer churn handler 亦會更新網路拓撲。透過改寫 PCL 生成演算法，可以實作不同的 Peer 加入演算法。反之，透過改寫 Peer churn handler，可以實作不同的 Peer 離開演算法。

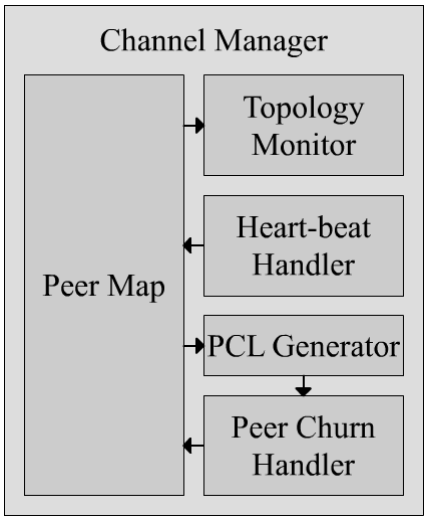


圖 36 Channel manager 的軟體模組圖

4. Database

Tracking server 採用 SQLite 作為資料庫，用來儲存使用者的登入資訊及 Peer 的運行資訊，以供後續的分析使用。

6.2.2 Streaming server

Streaming server 可向 Tracking server 註冊多個影音頻道。為了讓一個影音頻道能夠運作，Streaming server 將工作分成 5 個模組，分別是 Stream generator、Relay buffer、Stream relayer、Connection pool 及 Passive socket，如圖 37 所示。

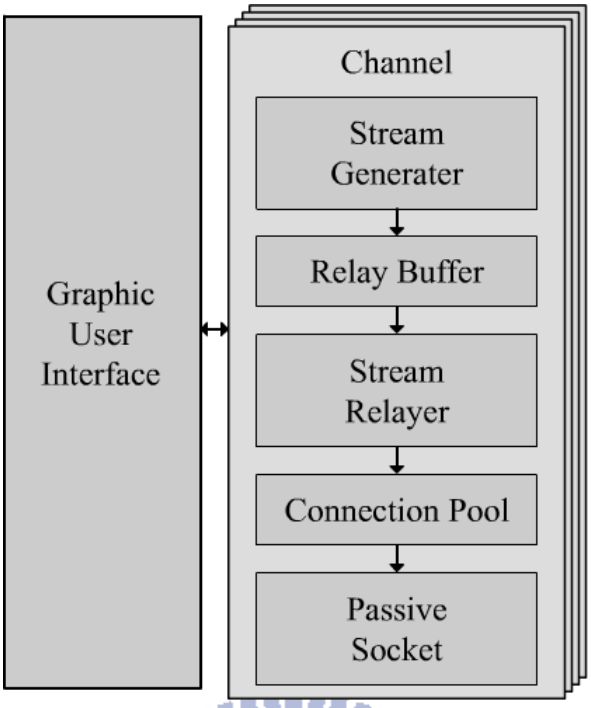


圖 37 Streaming server 的軟體模組圖

1. Stream Generator 及 Relay Buffer

Stream generator 透過 Xuggle API[33]，持續地從影音來源中取得影音資料，並將影音資料及相關播放資訊(如 Sequence number、Data type、Timestamp)封裝成自行定義的 DataPacket 封包，然後依序填入 Relay buffer 中。此外，若欲將程式控制訊息(如 end of stream)告知所有子節點，可將控制訊息封裝成自行定義的 ControlMessage 封包，然後填入 Relay buffer 即可。Relay buffer 是一個先進先出佇列(First-in-First-out queue, FIFO queue)，用來儲存將被轉送到所有子節點的 DataPacket 及 ControlMessage。

2. Passive Socket 及 Connection Pool

Streaming server 須在某特定的通訊埠上開啟一個 passive socket，用來等待來自 Peer 的連線建立請求。建立成功後的連線會被儲存到 Connection pool，該 pool 中儲存了與所有子節點間的網路連線對應，以便後續的管理與影音資料的轉送。

3. Stream Relayer

Stream relayer 負責將 Relay buffer 中的 DataPacket 及 ControlMessage，透過儲存在 Connection pool 中的網路連線對應轉送到所有的子節點。

6.2.3 Peer

Peer 主要由 8 個模組所組成，分別是 Relay buffer、Stream relayer、Connection pool、Passive socket、Video player、Playback buffer、Stream receiver 及 Active socket，如圖 38 所示。

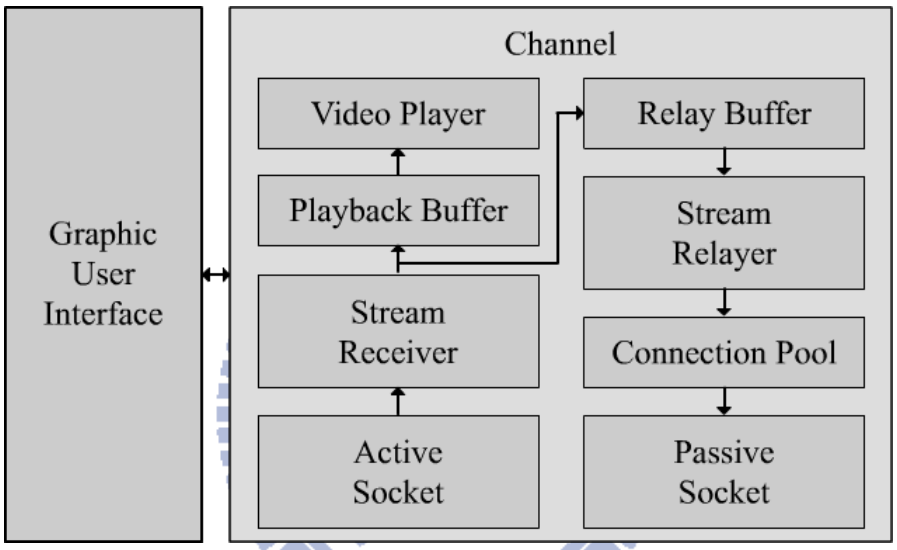


圖 38 Peer 的軟體模組圖

1. Video Player 及 Playback Buffer

將要被播放的影音資料會被填入 Playback buffer。Video player 會從 Playback buffer 中取出影音資料，並透過 Xuggle API 將影音資料解碼及播放，如圖 39 所示。

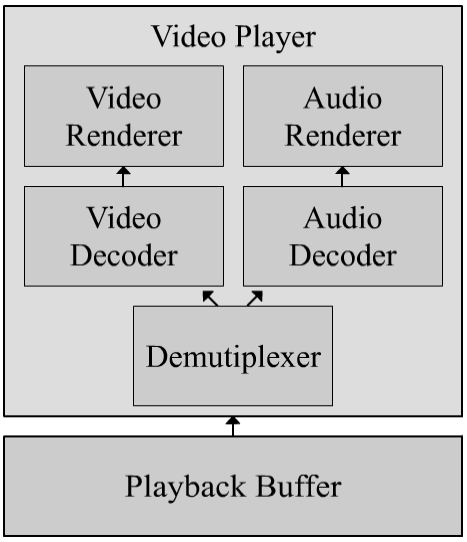


圖 39 Video player 軟體模組圖

2. Active Socket

從 Tracking server 取得 PCL 後，Peer 會從中選出適當的父節點，並主動和所選擇的父節點建立 active socket。

3. Stream Receiver

與父節點建立 active socket 後，父節點會將需要被轉送的資料透過 active socket 轉送給子節點。Stream receiver 負責接收來自父節點的資料，並將資料填入 Relay buffer 中。除此之外，所接收到的 DataPacket 會被複製一份到 Playback buffer 中，等待 Video player 播放。

4. Relay Buffer

Relay buffer 是一個先進先出佇列(First-in-First-out queue, FIFO queue)，用來儲存將被轉送到所有子節點的 DataPacket 及 ControlMessage。

5. Passive Socket 及 Connection Pool

Peer 須在某特定的 port 上開啟一個 Passive socket，用來等待來自其他 Peer 的連線建立請求。建立成功後的連線會被儲存到 Connection pool，該 Pool 中儲存了與

所有子節點間的連線對應，以便往後的管理與轉送資料。

6. Stream Relayer

Stream relayer 負責將儲存在 Relay buffer 中的 DataPacket 及 ControlMessage，透過 Connection pool 中的連線傳遞到所有的子節點。

6.3 系統運行畫面

6.3.1 Tracking server 運行畫面

由於節點會定期的將網路資訊回報給 Tracking server，因此系統管理員可以利用 Topology monitor 來檢視網路狀況。圖 40 為 Topology monitor 運行畫面，圖中顯示了某個影音頻道的樹狀網路拓撲，其中最上方的綠色方塊為樹根，即 Streaming server，其餘每個綠色方塊表示一個 Peer。此外，此雛型系統提供系統管理員以滑鼠拖拉的方式來改變樹狀網路的拓撲。

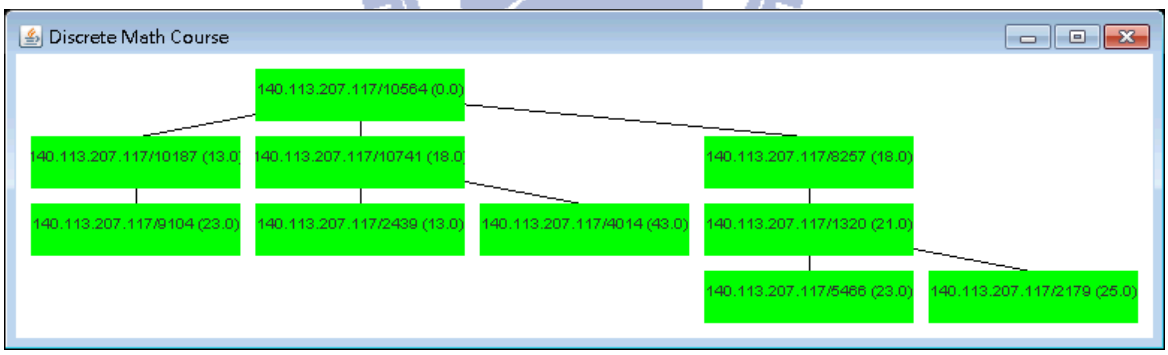


圖 40 Topology monitor 的運行畫面

6.3.2 Streaming server 運行畫面

Streaming server 可向 Tracking server 註冊新的影音頻道。如圖 41 所示，影音提供者指定好影音頻道的頻道名稱、敘述及影音來源後，只須按下 Start streaming 按鈕，就可完成頻道註冊並開始提供即時影音串流的服務。

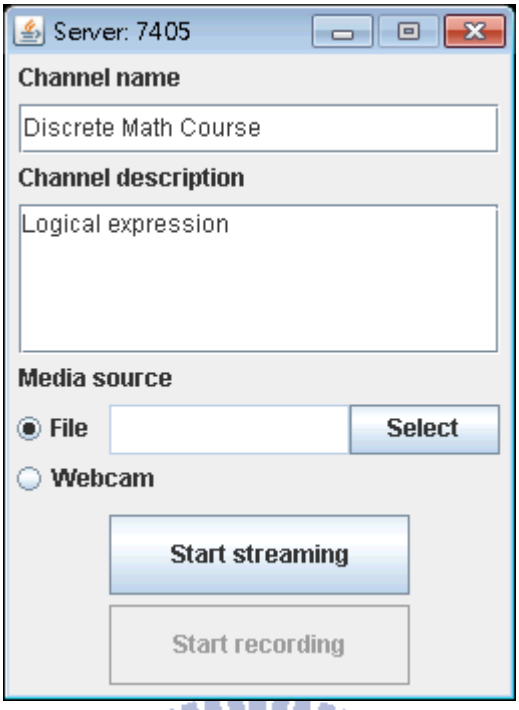


圖 41 Streaming server 運行畫面

6.3.3 Peer 運行畫面

如圖 42 所示，Peer 可從 Tracking server 取得最新的影音頻道清單、頻道敘述與影音串流截圖，使用者可以從中選擇其一來觀看即時影音串流。加入網路後，除了可看到即時影音串流外，Peer 的播放延遲時間及緩衝區資訊也會被即時的顯示出來。

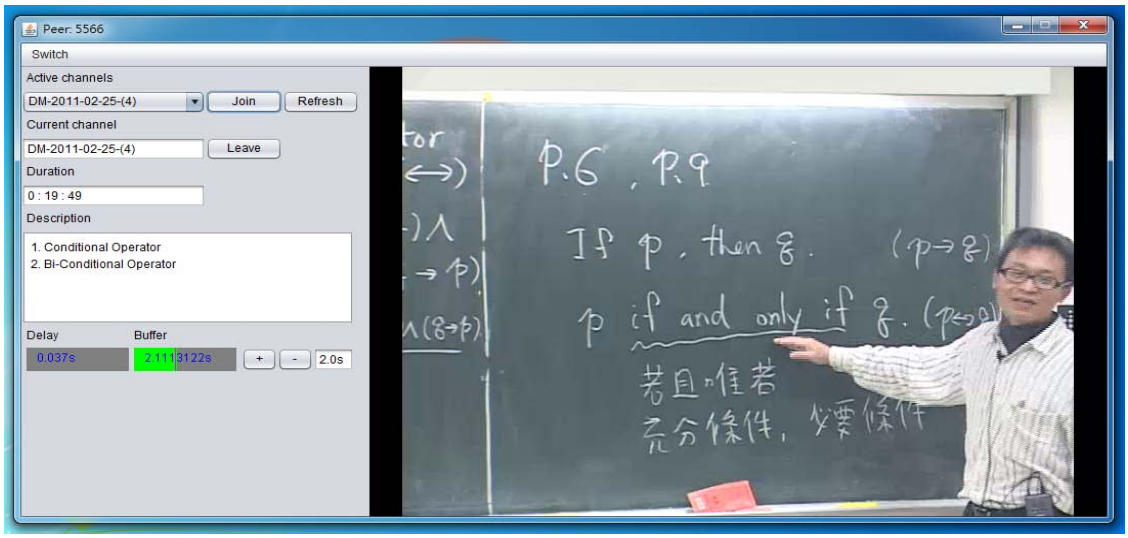


圖 42 Peer 的運行畫面

第七章、結論

7.1 討論

PPStream 及 PPTV 等商業軟體的崛起，證明了 P2P 技術除了可被應用於檔案分享服務外，也可被應用在隨選視訊及即時影音串流服務。本篇文章深入探討如何整合 P2P 技術來提供即時影音串流服務，尤其是專注於提升即時影音串流系統的系統容量。我們以樹狀 P2P 網路為背景，設計了兩個演算法，分別是 PCL 生成演算法及區域最佳化演算法。PCL 生成演算法讓新加入網路的節點能夠被分配到適當的樹狀網路位置，使得能貢獻較多上載頻寬的節點能夠經歷到較小的播放延遲時間。而區域最佳化演算法則提供系統一個分散式的方式來優化系統網路，用以降低系統平均播放延遲時間。模擬實驗結果顯示，相較於僅考慮播放延遲時間的演算法，我們所提出的演算法組合能獲得較佳的系統容量，特別是在提供高串流資料率與限制延遲時間上限時表現尤其突出。除了模擬工作以外，我們實作了一個基於樹狀 P2P 網路的即時影音串流系統雛型。該系統雛型讓我們能以真實網路環境來驗證與改進所提演算法，並已實際被應用在教學用途上。

7.2 未來工作

未來我們將以雛型系統為基礎，將所提的演算法組合加以實作並設計即時串流的緩衝與播放機制。再者，NAT 是所有 P2P 應用不可漠視的議題，除了專注提升系統容量外，我們也需提供 NAT traversal 機制，使得系統更臻完美。

參考文獻

[1] “Cisco Visual Networking Index: Forecast and Methodology, 2009-2014,” Cisco Systems, Inc, Technical Report, June 2010.

[2] Michael. O. Rabin, “Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance,” *Journal of the ACM*, vol. 36, pp. 335–348, April 1989.

[3] Valeria Cardellini, Michele Colajanni, and Philip. S. Yu, “Dynamic Load Balancing on Web-Server Systems,” *IEEE Internet Computing*, vol. 3, pp. 28–39, May 1999.

[4] Akamai, http://www.akamai.com/.

[5] “State of the Internet,” Akamai, Inc, Tech. Rep., 2010.

[6] Dropbox, http://www.dropbox.com/.

[7] Jochen Dinger and Hannes Hartenstein, “Defending the Sybil Attack in P2P Networks: Taxonomy, Challenges, and a Proposal for Self-Registration,” in *Proceedings of the 1st International Conference on Availability, Reliability and Security*, pp. 756–763, April 2006.

[8] Prithula Dhungel, Xiaojun Hei, Keith W. Ross, and Nitesh Saxena, “The Pollution Attack in P2P Live Video Streaming: Measurement Results and Defenses,” in *Proceedings of the Workshop on Peer-to-Peer Streaming and IP-TV*, pp. 323–328, August 2007.

[9] Matei Ripean, “Peer-to-Peer Architecture Case Study: Gnutella Network,” in *Proceedings of the 1st International Conference on Peer-to-Peer Computing*, pp. 99–100, August 2001.

[10] Bram Cohen, “Incentives Build Robustness in BiTtorrent,” in *Proceedings of the 1st*

Workshop on Economics of Peer-to-Peer Systems, May 2003.

[11] "Internet Study 2008/2009," ipoque GmbH, Tech. Rep.

[12] AeroFS, http://www.aerofs.com/.

[13] Jehan-Franscois Paris and Darrell D. E. Long, "A Proactive Implementation of Interactive Video-on-Demand," in *International Conference on Performance Computing and Communications*, April 2003.

[14] Yang Guo, Kyoungwon Suh, Jim Kurose, and Don Towsley, "P2Cast: Peer-to-Peer Patching Scheme for VoD Service," in *Proceedings of the 12th International Conference on World Wide Web*, pp. 301–309, May 2003.

[15] Siddhartha Annapureddy, Christos Gkantsidis, and Pablo Rodriguez, "Providing Video-on-Demand using Peer-to-Peer Networks," in *Proceedings of the Workshop on Internet Protocol TeleVision*, pp. 238–247, 2006.

[16] PPStream, http://www.ppstream.com/.

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

[18] Huey-Ing Liu and I-Feng Wu, "MeTree: A Contribution and Locality-Aware P2P Live Streaming Architecture," in *the 24th IEEE International Conference on Advanced Information Networking and Application*, pp. 1136–1143, April 2010.

[19] Xuping Tu, Hai Jin, Xiaofei Liao, and Jiannong Cao, "Nearcast: A Locality-Aware P2P Live Streaming Approach for Distance Education," in *ACM Transactions on Internet Technology*, vol. 8, pp. 2:1–2:23, February 2008.

[20] Xinyan Zhang, Jiangchun Liu, Bo Li, and Tak-Shing Peter Yum, "CoolStreaming/DONet: A Data-Driven Overlay Network for Peer-to-Peer Live Media Streaming," in

Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies, pp. 2102–2111, March 2005.

[21] Ayalvadi J. Ganesh, Anne-Marie Kermarrec, and Laurent Massoulie, "Peer-to-Peer Membership Management for Gossip-Based Protocols," in *IEEE Transactions on Computers*, vol. 52, pp. 139-149, February 2003.

[22] John Jannotti, David K. Gifford, Kirk L. Johnson, Frans M. Kaashoek, and James W. O'Toole, "Overcast: Reliable Multicasting with an Overlay Network," in *Proceedings of the 4th Conference on Symposium on Operating System Design Implementation*, pp. 197–212, October 2000.

[23] Duc A. Tran, Kien A. Hua, and Tai Do, "ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming," in *Proceedings of Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2, pp. 1283–1292, March 2003.

[24] Kunwoo Park, Sangheon Pack, and Taekyoung Kwon, "Climber: An Incentive-based Resilient Peer-to-Peer System for Live Streaming Services," in *Proceedings of the 7th International Conference on Peer-to-Peer Systems*, pp. 10–10, February 2008.

[25] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh, "SplitStream: High-bandwidth Content Distribution in a Cooperative Environment," in *the 2nd International Workshop on Peer-to-Peer Systems*, pp. 298-313, February 2003.

[26] Venkata N. Padmanabhan and Kunwadee Sripanidkulchai, "The Case for Cooperative Networking," in *Proceedings of the 1st International Workshop on Peer-to-Peer Systems*, March 2002.

[27] Feng Wang, Yongqiang Xiong, and Jiangchuan Liu, "mTreebone: A Hybrid Tree/Mesh

Overlay for Application-Layer Live Video Multicast," in *the 27th International Conference on Distributed Computing Systems*, pp. 49-49, June 2007.

[28] Tara Small, Ben Liang, and Bo Li, "Scaling Laws and Tradeoffs in Peer-to-Peer Live Media Streaming," in *Proceedings of the 14th Annual ACM International Conference on Multimedia*, pp. 539-548, July 2006.

[29] Chao Liang, Yong Liu, and Keith W. Ross, "Topology Optimization in Multi-Tree Based P2P Streaming System," in *21st IEEE International Conference on Tools with Artificial Intelligence*, pp. 806-813, November 2009.

[30] Michael R. Garey, David S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, ISBN 0-7167-1045-5. A2.1: ND1, p. 206.

[31] Xiaoming Zhou and Piet V. Mieghem, "Hopcount and E2E Delay: IPv6 Versus IPv4," in *the 6th International Workshop on Passive and Active Network Measurement*, vol. 3431, pp. 345–348, April 2005.

[32] 中華電信2010年營運報告, http://www.cht.com.tw/StockDownload.php?id=2226

[33] Xuggle, http://www.xuggle.com/.