國立交通大學

資訊學院資訊科技（IT）產業研發碩士班

碩 士 論 文

改善起始延遲及畫面品質之使用者經驗於使用可調式視訊編碼的點對點串流系統

Enhancing QoE of Start-up Latency and Picture Quality for P2P Streaming with Scalable Video Coding

研 究 生：陳玟瑾

指導教授：林盈達 教授

中 華 民 國 九 十 九 年 六 月

改善起始延遲及畫面品質之使用者經驗於使用可調式視訊編碼的點對點串流系統

Enhancing QoE of Start-up Latency and Picture Quality for P2P Streaming with Scalable Video Coding

研 究 生：陳玟瑾　　Student：Wen-Chin Chen

指導教授：林盈達　　Advisor：Dr. Ying-Dar Lin

國 立 交 通 大 學

資訊學院資訊科技（IT）產業研發碩士班

碩 士 論 文

A Thesis

Submitted to College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Industrial Technology R & D Master Program on
Computer Science and Engineering

June 2010

Hsinchu, Taiwan

中華民國九十九年六月

改善起始延遲及畫面品質之使用者經驗於使用可調式視訊編碼的點對點串流系統

學生：陳玟瑾　　指導教授：林盈達

國立交通大學資訊學院產業研發碩士班

摘　　要

傳統主從式(Client/Server)架構的串流技術易造成供應端負載過大、供應源的頻寬不足而使服務效能低落；而點對點 (Peer-to-Peer) 架構則藉由分散供應源以減輕主伺服器之負擔，大為改善主從式架構所遭遇的問題，因此點對點串流系統越來越風行。但點對點串流技術的相關研究大多著重於挑選最佳下載節點、節點分布最佳化、群組建構等點對點通訊協定之演算法，少有文獻特別針對改善使用者經驗(Quality of Experience，QoE)之方法討論，像縮短等待時間及提升觀看畫質等。當系統效能再高若使用者經驗過差也無法提升其使用率。因此對於串流系統，使用者觀看時的感受也應該被重視。同時，不同使用者的使用需求也不甚相同，因此我們在點對點網路環境下利用可調式視訊編碼技術(Scalable Video Coding)的分層影像品質特性，改善起始延遲以及畫面品質之使用者經驗，讓使用者在觀看串流影像時有最小的延遲時間並能享受到最佳的畫面品質。

本論文提出一 Layer Deadline Scheduling(LDS)演算法，藉由計算各階層的最終播放時間排程下載順序。若實際下載時間與預測衝突時，便以優先權值決定下載或放棄。實驗結果顯示，比較四種演算法：(1) LDS Algorithm、(2) Random Selection Algorithm、(3) Smooth Algorithm without delay、(4) Smooth Algorithm with 1 sec delay，當網路環境的 RTT/2 從 0 ms 增加到 300 ms 時，LDS 演算法表現最佳。畫面品質更是 Smooth Algorithm 的兩倍以上，而有效的下載率更可達 80%以上。

Enhancing QoE of Start-up Latency and Picture Quality for P2P Streaming with Scalable Video Coding

student：Wen-Chin Chen Advisor：Dr. Ying-Dar Lin

Industrial Technology R & D Master Program of
Computer Science College
National Chiao Tung University

ABSTRACT

Client/Server streaming is susceptible to server's overloading and liable to exhausting the usage of server's bandwidth; hence, server's service is degraded. The above deficiencies can be solved by P2P streaming where a resource provider is split and distributed to multiple nodes over a network, so the server's loading can be mitigated. Therefore, P2P streaming becomes prevailing in recent years. Most of the researchers focus on peer selection, network structure, group organization, and so on. The issue of improving Quality of Experience (QoE) which is used to specify user's perception is scarcely discussed. User waiting time and picture quality are two of the metrics for QoE. If QoE is low, no matter how system performance is, the usage rate cannot rise. Therefore, the metric of QoE must be taken into account by a streaming provider. But different users have different usage demands, the layered quality characteristic of Scalable Video Coding (SVC) are adopted to enhance the QoE. This will lead to the best picture quality and the least user wait time for users.

In this study, we propose a Layer Deadline Scheduling (LDS) Algorithm to schedule all layers by their playback deadline and calculate the current available bandwidth to make sure if the streaming file being downloaded can be received in time. If not, whether to download a streaming file or skip it depends on its priority. A comparison of the empirical results of four algorithms: (1) LDS Algorithm, (2) Random Selection Algorithm, (3) Smooth Algorithm without delay, and (4) Smooth Algorithm with 1 sec delay, shows when RTT/2 ranges from 0 to 300 ms, LDS Algorithm has the best QoE. The picture quality with LDS Algorithm doubles than that with Smooth Algorithm. Moreover, the proportion of discarded layers with LDS Algorithm is as low as 10% of all files.

Table of Contents

摘　　要 i
ABSTRACT ii
Table of Contents iii
List of Tables iv
List of Figures v
1 Introduction 1
2 Background 4
2.1 P2P Architecture and Streaming 4
2.2 Scalable Video Coding 7
2.3 Other Related Works 8
3 Problem Statement 10
3.1 Parameter Definition 10
3.2 Problem Definition 11
4 Layer Deadline Scheduling Algorithm 12
4.1 Architecture Overview 12
4.2 Selection Phase 13
4.3 Scheduling Phase 14
4.4 Example Run 17
5 System Implementation 18
5.1 Pre-Processing 18
5.2 LDS Algorithm 19
6 Experiments and Results 21
6.1 Test Environment 21
6.2 Test Cases 21
6.3 Test Results 22
7 Conclusions and Future Works 28
Reference 29

List of Tables

TABLE 1 A COMPARISON OF FOUR TYPES OF P2P ARCHITECTURES. 5
TABLE 2 A COMPARISON OF THE STRUCTURES OF P2P ARCHITECTURES. 6
TABLE 3 A COMPARISON OF P2P STREAMING SOFTWARES. 7
TABLE 4 A COMPARISON OF VIDEO CODING BETWEEN SVC AND MDC. 8
TABLE 5 VARIABLE NOTATION. 10
TABLE 6 VARIABLE NOTATION. 13
TABLE 7 THREE CASES OF LDS. 16
TABLE 8 THE NUMBER OF CHUNKS PER ROUND. 22
TABLE 9 THE COMPARISON OF RESULTS. 28

List of Figures

FIGURE 1 ZIGZAG PATTERN. 3
FIGURE 2 A CENTRALIZED DIRECTORY ARCHITECTURE. 4
FIGURE 3 A HIERARCHICAL OVERLAY ARCHITECTURE. 5
FIGURE 4 AN OVERVIEW OF A CHORD 6
FIGURE 5 A SIMPLIFIED DIAGRAM USING SVC. 8
FIGURE 6 AN OVERVIEW OF STREAMING LAYERS 10
FIGURE 7 AN OVERVIEW OF THE LAYER DEADLINE SCHEDULING ALGORITHM. 12
FIGURE 8 SELECTION PHASE. 14
FIGURE 9 THE PSEUDO CODE OF SELECTION PHASE. 14
FIGURE 10 A CONCEPTUAL DIAGRAM OF LDS ALGORITHM. 15
FIGURE 11 THE FLOW CHART OF LDS. 16
FIGURE 12 THE EXAMPLE RUN OF LDS 17
FIGURE 13 AN OVERVIEW OF SYSTEM IMPLEMENTATION. 18
FIGURE 14 AN OVERVIEW OF PRE-PROCESSING. 18
FIGURE 15 AN OVERVIEW OF LDS PROCEDURE. 20
FIGURE 16 TEST ENVIRONMENT. 21
FIGURE 17 PICTURE QUALITY. 23
FIGURE 18 WAITING TIME AND PICTURE QUALITY. 24
FIGURE 19 THE PERFORMANCE OF ALGORITHMS 25
FIGURE 20 USER WAITING TIME. 25
FIGURE 21 THE DISTRIBUTION FOR EACH CHUNK. 26
FIGURE 22 STARTUP LATENCY. 27
FIGURE 23 PICTURE QUALITY VS. USER WAITING TIME. 27

1 Introduction

Peer-to-Peer(P2P)共享技術的崛起

傳統主從式(Client/Server)架構下的視訊串流技術，像是 Youtube 網站，由於所有的使用者都向同一個服務端提出要求，造成服務端的負擔過重，使得正在觀看的使用者感受到因服務端繁忙而造成的延遲現象。同時，服務端的儲存設備以及頻寬能力也因成本考量的限制而僅能提供基本畫面品質的視訊內容，讓觀看的使用者即使擁有高頻寬也只能接收基本畫面品質的視訊內容。然而頻寬的限制，造成網路傳輸緩慢而必須讓使用者等待一段時間。為了改善傳統主從式架構的種種瓶頸，利用 P2P 共享技術的 P2P 視訊串流技術逐漸嶄露頭角，較為人所知的應用程式如 PPStream，PPLive。P2P 技術以資源共享為出發點，依賴網路系統中參與者分享的運算能力和頻寬，而非聚集在單一服務端，藉此達到分散服務端負擔的目的。

何謂使用者的經驗品質 QoE

然而目前的 P2P 視訊串流技術仍然存在一些研究議題被討論，例如如何動態管理節點、如何挑選最佳節點、特殊節點的獎勵機制等，其中最為人所探討的是與使用者息息相關的使用者經驗品質 Quality of Experience (QoE) [1]。QoE 泛指所有使用者在使用經驗上的任何感覺，例如：畫面的流暢度與品質或是等待視訊影像的時間。QoE 通常可用視頻平均主觀評分(Mean Opinion Score for Video，MOS_V)的分數 1~5 分來評斷，分數越接近 5 分表示 QoE 的品質越好。服務提供者可以依據 QoE 的結果來了解需要在哪些地方做改進以滿足使用者的最高需求。與 QoE 相關的是較為人知的 QoS (Quality of Service)，QoS 是以客觀的測量目標來表達服務系統能力的好壞，像是系統吞吐率、封包遺失率、延遲率、或是錯誤率等，使用者可依據這些項目的數值來了解服務系統的優劣。QoE 和 QoS 最大的差異在於 QoE 考慮到的是使用者的使用感受，這也是當今社會所重視的議題，

因此本篇論文著重如何提升使用者的 QoE。

滿足 P2P 異質性的可調式視訊編碼 (Scalable Video Coding)[2]

高 QoE 的 P2P 視訊串流服務具備以下幾個特點：(1)起始影像緩衝時間短、(2)使用者觀看時的中途等待時間短，以及(3)流暢且高畫質的畫面[3]。要縮短影像緩衝時間以及使用者等待時間就必須考慮不同的 P2P 網路的運作架構。不同的 P2P 架構影響了搜尋檔案來源的速度，像是 Gnutella，透過大量的廣播訊息來尋找某一節點。除了廣播訊息造成網路中的非必要流量增加，若要尋找的節點存在，卻可能因為距離太過遙遠而造成搜尋失敗。若是要尋找的節點不存在，也必須等到 timeout 時才會知道，造成了這段時間的浪費。然而由於 P2P 網路架構中的每個節點都不同的網路能力以及使用需求，因此若要提供流暢且高畫質的畫面可考慮階層式編碼方式(Layered Coding)，例如：可調式視訊編碼(Scalable Video Coding，SVC)。SVC 將一個影像內容壓縮成多個階層(Layer)的影像資料，並依取得階層的數量來呈現不同的畫面品質。每個節點可依照其網路能力及需求取得不同數量的階層。若某一節點的網路能力很好，則他可取得多個階層來增加影像的品質。反之，當網路能力不好時，便取得基本的階層僅以基本的影像品質為訴求。

將 SVC 作為 P2P 串流技術的編碼方式的領域裡，在縮短緩衝時間的部分，Mushtaq 等人[4][5]提出了 Hybrid Overlay Management[4]，將不同的階層檔案來源各成一套重疊網路，並從各自的重疊網路中選出最佳的點。以及 Packet Scheduling Mechanism[5]，從候選節點清單中挑選最佳的一個節點來要求最重要的 layer 檔案，次佳的節點要求次重要的檔案，藉此來縮短緩衝時間。Lee 等人[6]則是提出一套 GaiaSharp 系統，藉由動態調整 TCP 協定的訊框大小(Window Size)來改變要求階層檔案的順序，因而改善了使用者的等待時間。除此之外，Mushtaq 等人[4][5] 還利用了如圖 1 的 *ZigZag Pattern*，一種 Z 字形的順序，調整要求 SVC 階層檔案的順序，來改善了畫面品質。

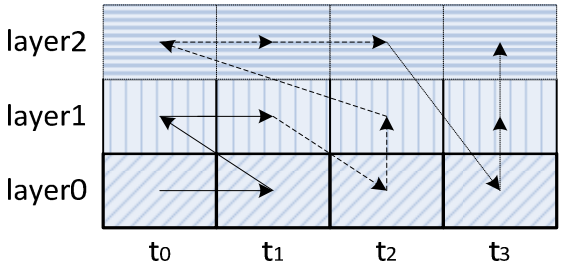


Figure 1　ZigZag Pattern.

在這篇論文裡，我們利用 SVC 的特性：針對不同階層檔案的重要性以及最終播放時間，提出了一個排程演算法，Layer Deadline Scheduling (LDS)。在結構化 P2P 的架構中，利用雜湊演算法縮短搜尋節點的時間。再根據每個階層檔案的最終播放時間來排定下載順序，並參考其重要性來進行下載順序的調整，達到減少緩衝時間以及使用者等待時間，並提供最好的畫面品質的目的，進而改善了使用者的 QoE。

最後使用了 JSVM 6.8.2[7]以及類似 Chord 的 P2P 架構來完成系統的實作。JSVM 是一套用來將完整的視訊串流壓縮成多個階層檔案的工具。首先利用了 JSVM 將目標串流檔案壓縮成 3 個階層檔案，再將這 3 個階層檔案分散至 Chord 系統中。當某個節點提出要求時，便利用這套系統搜尋相關內容並且下載，最後解碼回完整的串流視訊。我們進行了關於起始延遲(Startup Latency)、使用者等待時間(User Waiting Time)、畫面品質(Picture Quality)、和演算法效能(Performance of Algorithm)的測量與評估。並提出實驗的觀察結果。

本文章節安排如下：第二章介紹了 P2P 視訊串流技術和 SVC 的背景，以及相關研究。第三章定義了 Problem Statement。在第四章提出了一套 Layer Deadline Scheduling (LDS)的排程演算法來改善使用者的 QoE。第五章描述了實現 LDS 演算法的實作系統。第六章整理出實驗的觀察結果。最後在第七章則對本文作總結。

2 Background

不同的 P2P 架構由於不同的節點建構方式使得搜尋的方法也不盡相同，造成搜尋某一特定節點所花費的時間也因此不同，然而目前的這些 P2P 架構其優缺點為何？另外，當複雜的 P2P 網路環境中不同使用者的需求變異大時，早期單一階層的影像壓縮方式是無法滿足多樣化的使用者需求，因此便出現結合可調式視訊編碼(SVC)在 P2P 網路的應用產生，然而同時考慮 SVC 的壓縮特性與 P2P 網路架構時會出現哪些問題？又有哪些先前討論的解法呢？此章節將逐一介紹相關背景知識及以上問題。

2.1 P2P Architecture and Streaming

P2P Architecture

目前的 P2P 架構可依網路中分享節點的配置的方式分為四類：

(1) 集中式檔案架構(Centralized Directory)[8]，如下圖 2，由一個中央檔案系統 S 管理網路中所有的節點，因此只要該節點存在就保證會被找到。但當節點 C 離線時，節點 D、E 就失去了與系統的連繫，因此存在單一節點失效的問題。

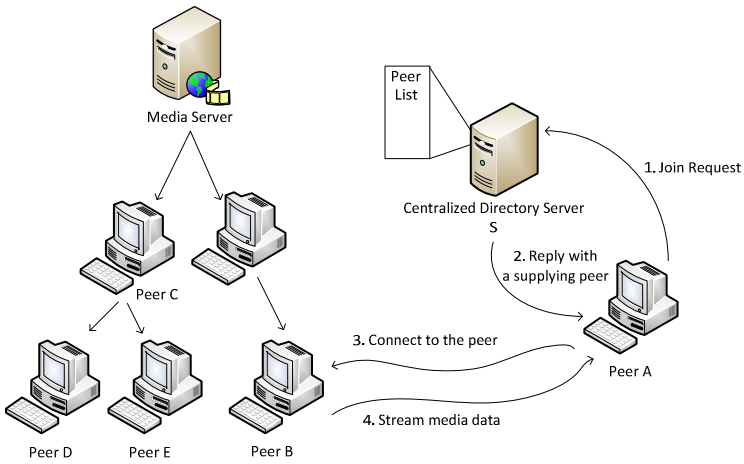


Figure 2 A Centralized Directory Architecture.

(2) 階層式重疊架構(Hierarchical Overlay)[8]，由不同的重疊網路組成。每一個重疊網路都提供一個檔案來源，並將相同的網路能力的節點歸類為同一階層，如下圖 3。如此一來不僅保障了節點的搜尋，也改善了單一節點失效的問題，但

缺點是實作不易。

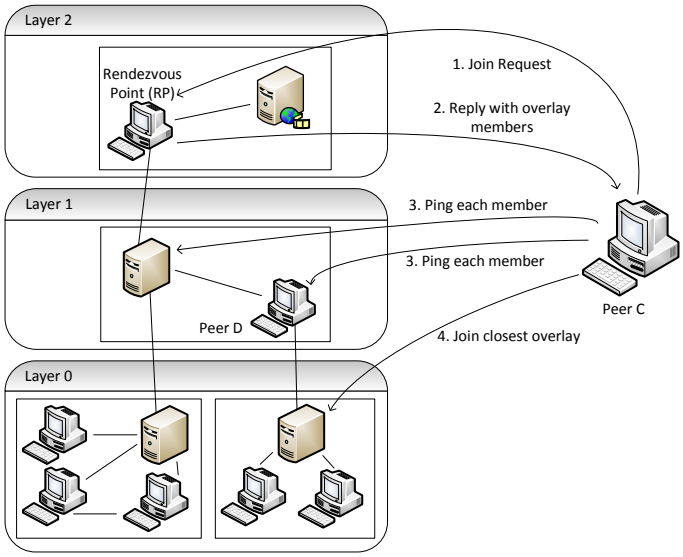


Figure 3　A Hierarchical Overlay Architecture.

(3) Controlled flooding[8]，在這種 P2P 架構之下，不存在中央的管理者，因此要搜尋某一節點必須透過大量的廣播訊息傳送，造成網路系統中有過多非必要的流量。通常廣播訊息會設定 Time-to-Live(TTL)值以避免等待時間過長，當某一存在的節點距離超過 TTL 時，將無法被找到。

(4) 結構化 P2P 架構(Structured P2P)[8]，網路系統中所有的節點由一特定演算法所維護，像是分散雜湊表(Distributed Hash Table，DHT)。當要尋找某一節點時，可透過該演算法快速的找到存在的節點，此舉不僅保證了搜尋也解決了單一節點失效的問題。

下表 1 整理出上述四種 P2P 架構之比較，其中結構化 P2P 架構不僅提供了高可靠度以及搜尋的便利性，其實作也較階層式重疊架構容易。

Table 1　A Comparison of Four Types of P2P Architectures.

Approaches	Centralized Directory	Hierarchical Overlay	Controlled flooding	Structure P2P
Scalability	Low	High	Medium	High
Single Point of Failure	Yes	None	None	None
Search Guarantee	Yes	Yes	None	Yes
Implementation	Simple	Difficult	Medium	Medium

而表 2 則列出四種常見結構化 P2P 架構的相關比較。

Table 2　A Comparison of the Structures of P2P Architectures.

	Chord	Pastry	Tapestry	CAN
Architecture	Circle ring	Circle ring	Suffix matching	Multi-dimensional coordinates space
Per node	Node ID	Node ID	Node ID	Multi-dimensional space
Per file	Object ID	Object ID	Object ID	Coordinate
Support	Finger table	Routing table	-	Coordinate routing table
Compare files	Jump search	Two level search	Matching suffix	Greedy algorithm

由表 2 可知，Chord 和 Pastry 都將所有節點經由雜湊函數(Hash Function)給予一個識別的節點代碼，這些節點便會依其代碼位置形成一個邏輯環，如下圖 4。當有檔案要加入時，也會將其檔名進行雜湊，並交由鄰近的節點代碼進行保管。為了加快搜尋的速度，Chord 中的每個節點各自維護了一個路由表(Finger Table)，透過跳躍式的比對路由表中的節點代碼便可快速的找到檔案的資訊。Pastry 和 Tapestry 在比對檔案方面較 Chord 複雜；而 CAN 則是建構於一座標平面上。

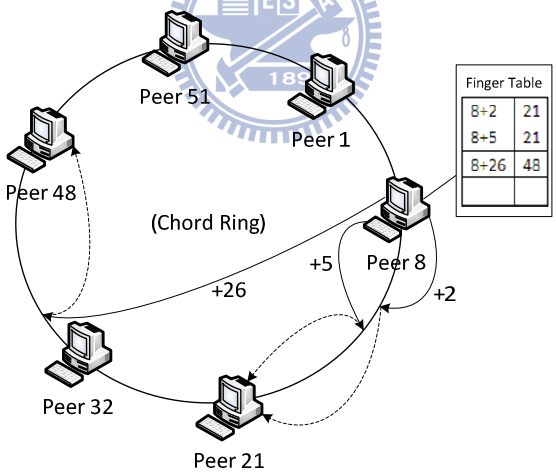


Figure 4　An Overview of a Chord.

P2P Streaming

P2P 視訊串流是近期發展很成功的 P2P 應用。自從 2005 年 Li 等人[9]提出 DONet 的概念並實作出 CoolStreaming 後，P2P 串流軟體便如雨後春筍般的相繼出現，像是 PPLive[10]、PPStream[11]、TVAnts[12]和近期的 Joost[13]。下表 3 將目前較為人所知的 P2P 串流軟體做簡單的比較。其中在起始等待時間方面，PPLive 和 PPStream 在開始時須要等待 15 秒的廣告時間以緩衝影像，而 TVAnts

則是依串流內容的熱門度而有所不同。Joost 是 2007 年才由 Skype 和 Gnutella 創辦人共同提出，建立於 Content Delivery Network (CDN)系統。由於 Joost 所提供的串流內容都不長，因此其影像緩衝時間只需要兩秒左右。

Table 3　A Comparison of P2P Streaming Softwares.

	PPLive	PPStream	TVAnts	Joost
Start time	Feb. 2005	Jan. 2006	May. 2006	Jan. 2007
Viewing tool	Client program	Client program	Client program	Web viewer
Content service type	Video On Demand	Video On Demand	On Live	Video On Demand
Startup latency	15s(ads)	15s (ads)	12s~70s	2s

Related Works of P2P Streaming

現在仍然有許多的研究在探討 P2P 串流的相關議題，像是負載平衡(Load Balancing)[14][15]、片段選擇(Piece Selection)[16][17]、低延遲(Low Delay)[14][15][16][17] 等。延遲的現象通常發生在某個時間點該影像來不及取得而無法順利播放，而為了達到低延遲的目的，有些研究者利用可調式視訊編碼(SVC)的特性，當缺少的部分來不及取得時依然可以低品質的影像來播放，來減少延遲現象的發生。

2.2　Scalable Video Coding

可調式視訊編碼(SVC) 除了具有傳統 H.264/AVC 的高編碼效率優點外，更提升了編碼的彈性。主要的原因為 SVC 由一個基礎層(Base Layer)及數個增強層(Enhancement Layer)所構成，擁有時間可調性(Temporal Scalability)、空間可調性(Spatial Scalability)及訊雜比可調性 (SNR Scalability)三大特性。其中基礎層的編碼方式類似於 H.264/AVC，而增強層除了可自行做預測與編碼外，亦利用基礎層之編碼資訊進行預測與編碼。下圖 5 為簡易示意圖，主伺服器提供一份多層次的串流影像，可依接收使用者的不同傳輸需求，傳送不同數量的階層影像。

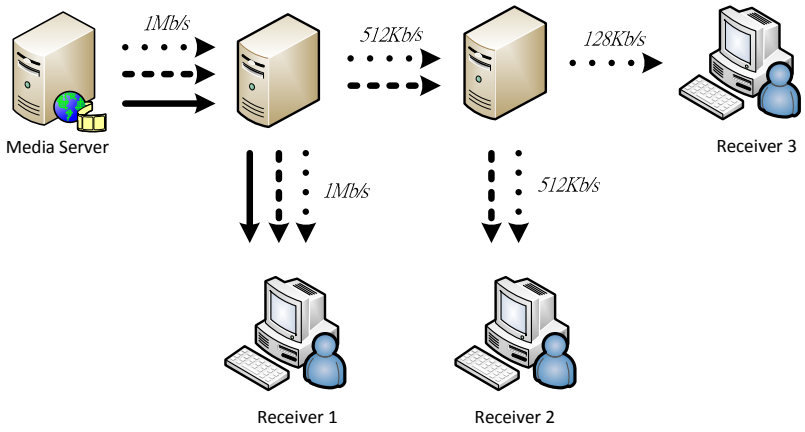


Figure 5　A Simplified Diagram Using SVC.

另外，還有另一種編碼方式：多重描述編碼(Multiple Description Coding，MDC)。它是以水平的時間間隔將影像切割成多個描述片段，每個描述片段可以獨立編碼，但每個描述片段所呈現的畫面品質都是一樣的。若有遺失的描述片段則延遲現象便會發生。如表 4 所示 SVC 對於多樣化的使用者來說能提供最有彈性的串流影像內容。

Table 4　A Comparison of Video Coding between SVC and MDC.

	SVC (Scalable Video Coding)	**MDC (Multiple Description Coding)**
Organization	Joint Video Team (JVT)	
Unit	Per layer (a base layer and enhancement layers)	Per description
Type of decoding	Based on a base layer and previous enhancement layers	Independent
Files obtained	Dependent on bandwidth	The same description
Rate Scalable	Large	Small
Packet Loss Effect	Picture quality decrease	Playing delay

2.3　Other Related Works

Mushtaq 等人[9]提出一套在 Gnutella 下的混合式重疊網路的管理機制(Hybrid Overlay Network Management)。這套系統由多個重疊網路所組成，每個重疊網路皆提供同一份階層內容(Layer)。在同一個重疊網路裡將所有的節點依其單一來回時間(Round Trip Time，RTT)建構成一個最小堆疊(MinHeap)。因此直接向每個重

疊網路的根結點(Root)取得該階層內容便可縮短下載時間。而後他們又再提出一個流暢視訊發展(Smooth Video Delivery)的方法[5]。跟前一個研究相似的是，藉由改變 SVC 每個階層的要求順序來取得畫面品質與延遲時間的平衡。這個順序是一種 Z 字形的順序(ZigZag Pattern，如圖 1)。 但在這份研究中，他只建構了一個重疊網路而已，並從這個重疊網路中向效能最佳的一個節點提出基礎層的要求，向次佳的節點提出增強層的要求，以此類推。要將下載回來的檔案進行解碼時先檢查檔案間的相依性，若是基礎層不存在，則將其他的增強層都刪除，因為沒有基礎層就無法進行影像的解碼。最後 Lee 等人[6]在 2008 年實作了一個 P2P 視訊系統，GaisSharp。他以一套類似 BT 的搜尋系統，MonoTorrent，並藉由動態的調整 TCP 協定的訊框大小以縮短延遲時間。

3 Problem Statement

3.1 Parameter Definition

對於一個完整的串流影片而言，如圖 6 所示，首先依照固定的時間間隔切分成多個區塊(chunk)，再依照 Scalable Video Coding 的方式將每一個區塊壓縮成多個階層(layer)。通常會有一個基礎層以及一個以上的增強層。為了說明方便，這裡統一稱基礎層為 *Layer 0*，增強層 *1* 為 *Layer 1*，以此類推，增強層 *M* 為 *Layer M*。在本文中將視每一個區塊的每一個階層為獨立的個體來討論之。

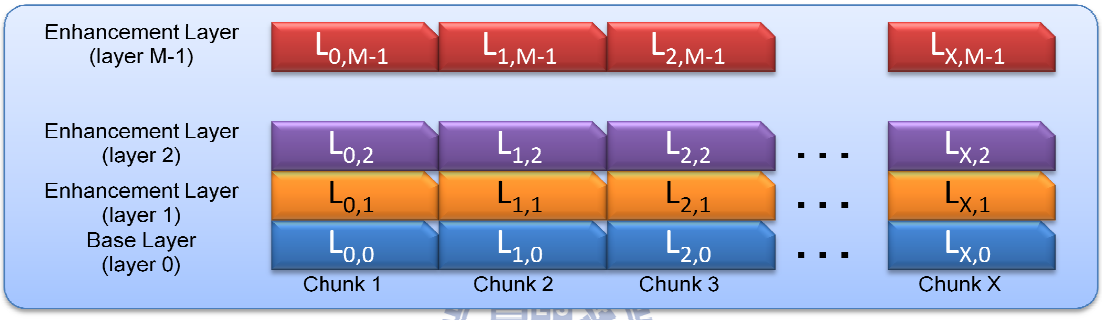


Figure 6 An Overview of Streaming Layers.

為了後續章節能精確的描述問題與討論，我們定義了九個變數符號如下表 5 所示：M、N為已知分別代表階層數量以及區塊數量，其中我們為了達到批次處理的目的，以N個區塊為單位進行每一回合(Round)的程序；$L_{i,j}$用來表示第 i 個區塊的第j個階層之物件(Object)；而$C_{i,j}$則表示$L_{i,j}$在系統中的候選節點數量，D_i表示第 i 個區塊的最終播放時間，只要在D_i之前取得第 i 個區塊就沒有延遲；$S_{i,j}$代表$L_{i,j}$的大小，可透過(1)與目前的可用頻寬BW得出$L_{i,j}$所需的傳輸時間$TR_{i,j}{}'$；另外TC_i則用來表示第 i 個區塊完整下載的時間，也就是可以播放的時間。

$$TR_{i,j}{}' = S_{i,j}/BW \tag{1}$$

Table 5 Variable Notation.

Notation	Description
M	The number of layers for each chunk.
N	The number of chunks per round
D_i	The playback deadline of chunk i

$L_{i,j}$	The object of layer *j* of chunk *i*
$C_{i,j}$	The number of $L_{i,j}$'s candidate peers in the system
$S_{i,j}$	The size of $L_{i,j}$
BW	The current available bandwidth.
$TR_{i,j}{}'$	The real transmission time of $L_{i,j}$, as shown in (1).
TC_i	The complete download time of chunk *i*

3.2 Problem Definition

當：(1)一段串流影像的每一個區塊，都被分成M個階層，並且(2)對於每一個不同的$L_{i,j}$，都有$C_{i,j}$個候選的節點可供下載。

為了實現(1)避免使用者在觀賞的期間因發生播放延遲而必須等待以及(2)有較好的畫面品質兩項需求，則

$$\forall i, i = 0..N-1\,;\ \forall j, j = 0..M-1$$

從$C_{i,j}$個候選的節點中挑選一個最佳的節點下載$L_{i,j}$以滿足：

$$min\,(TR_{i,j}{}') = S_{i,j}/BW \ \ s.t.$$

$$\begin{cases} a.\,if\ TC_i > D_i,\ TC_i - D_i \ => \ \ min \sum_{i=0}^{N-1} TC_i - D_i \\ b.\,max\{Num(L_{i,j})\} \end{cases}$$

藉由$L_{i,j\in 0..M-1}$的實際傳輸時間$TR_{i,j\in 0..M-1}{}'$而取得區塊i的下載完成時間TC_i，當TC_i超過區塊 i 原本的最終播放時間D_i時，這個時間差就稱為等待時間，為了改善 QoE 就必須使得這個時間差的總和為最小值，而讓使用者等待時間為最短；另外若要改善整體的畫面品質，則對每一個區塊而言必須取得最多數量的階層，因此如何在每個區塊 i 的最終播放時間D_i之前完成最多階層數的傳輸? 是這篇論文所要探討的問題。

4 Layer Deadline Scheduling Algorithm

為了在每個區塊 i 的最終播放時間D_i之前完成最多階層數的傳輸，我們將使用者送出觀看需求到實際收到影像的這段時間內，依先後順序分為三個階段：搜尋階段(Searching Phase)、選擇階段(Selection Phase)、排程階段(Scheduling Phase)，並藉由最佳化各個階段來達成目標。

4.1 Architecture Overview

Layer Deadline Scheduling 演算法分成三個階段，搜尋階段、選擇階段以及排程階段，如下圖 7 所示。

「搜尋階段」透過類似 Chord 的 P2P 系統，利用 DHT 跳躍式的尋找檔案所存在的節點，縮短尋找時間。

「選擇階段」從候選節點清單裡挑出一個最適合的節點 x，節點 x 必須擁有最小的來回時間(RTT)，表示跟節點 x 的傳輸會最快，藉此縮短下載的傳輸時間。

「排程階段」將針對所有需要傳輸的階層(Layer)作傳輸順序的安排，以達到「在期限內取得最多所需階層」的目標。

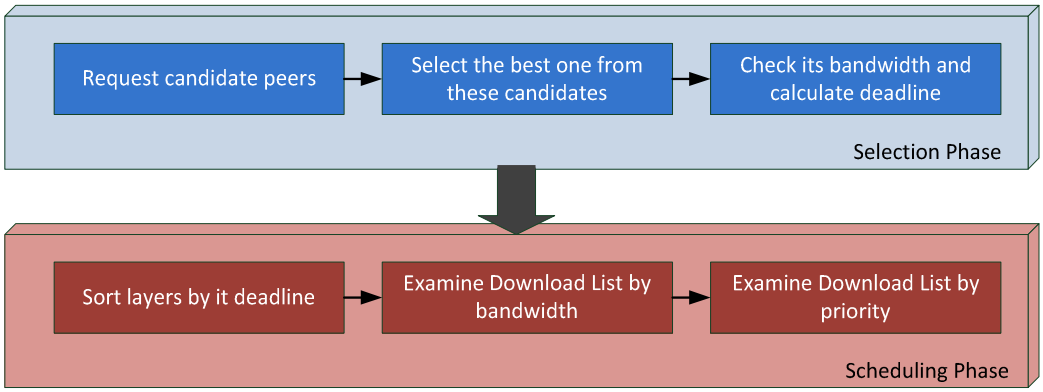


Figure 7　An Overview of the Layer Deadline Scheduling Algorithm.

在說明 LDS Algorithm 之前，先定義表 6 的六組符號，以便後續敘述精確。對每一個$L_{i,j}$來說都有一個最佳節點$B_{i,j}$，在$C_{i,j}$個候選節點的清單中這個節點有最小的 RTT 值$RTT^*_{i,j}$，表示傳輸時間$TR_{i,j}$將會最短；而RTT_k則用來表示節點 k 的 RTT 值；根據 SVC 的特性，每個$L_{i,j}$都有一個最低頻寬需求$BW_{i,j}$的限制，當

某個節點的網路能力到達此需求時才有資格取得$L_{i,j}$；因此$L_{i,j}$的傳輸時間$TR_{i,j}$與最終播放時間$TD_{i,j}$便可分別依(2)及(3)求得；

$$TR_{i,j} = S_{i,j}/BW_{i,j} \quad (2)$$

$$TD_{i,j} = D_i - TR_{i,j} \quad (3)$$

最後$L_{i,j}$的優先權值$P_{i,j}$用來裁決當網路頻寬小於$L_{i,j}$的頻寬需求時的取捨，如(4)，*Layer 0* 的優先權值最高，為 *M-1*；*Layer M-1* 的優先權值最小，為 *0*。

$$P_{i,j} = M - 1 - j, \quad for\ j = 0 \sim M - 1 \quad (4)$$

Table 6　Variable Notation.

Notation	Description
$B_{i,j}$	The best peer of $L_{i,j}$.
RTT_k	Peer k's value of RTT. $RTT^*_{i,j}$ is $B_{i,j}$'s value of RTT.
$BW_{i,j}$	The required bandwidth of $L_{i,j}$
$TR_{i,j}$	The transmission time of$L_{i,j}$, as shown in (2).
$TD_{i,j}$	The download deadline of$L_{i,j}$, as shown in (3).
$P_{i,j}$	The priority of $L_{i,j}$. As (4) show.

4.2　Selection Phase

為了縮短下載的時間，我們針對每個要下載的$L_{i,j}$挑出一個最佳的節點$B_{i,j}$，這個節點擁有最小的 RTT 值$RTT^*_{i,j}$，如下圖 8。

以節點 A 為例，一開始先透過起始點(Bootstrap)取得候選節點清單(Candidate Peer List)，並得知這些候選節點擁有所需要的階層檔案。為了從這些候選節點中挑選出一個最佳的節點取得階層檔案，節點 A 便會送 Ping 封包給候選節點清單上的每一個節點，並選擇具最小 RTT 值的節點 D 為最佳節點。並檢查與節點 D 間的可用頻寬是否符合$L_{i,j}$所需要的傳輸頻寬$BW_{i,j}$，若符合則利用(3)計算出$L_{i,j}$的最終播放時間$TD_{i,j}$；若不符合，則從候選節點清單中挑選次佳的節點。

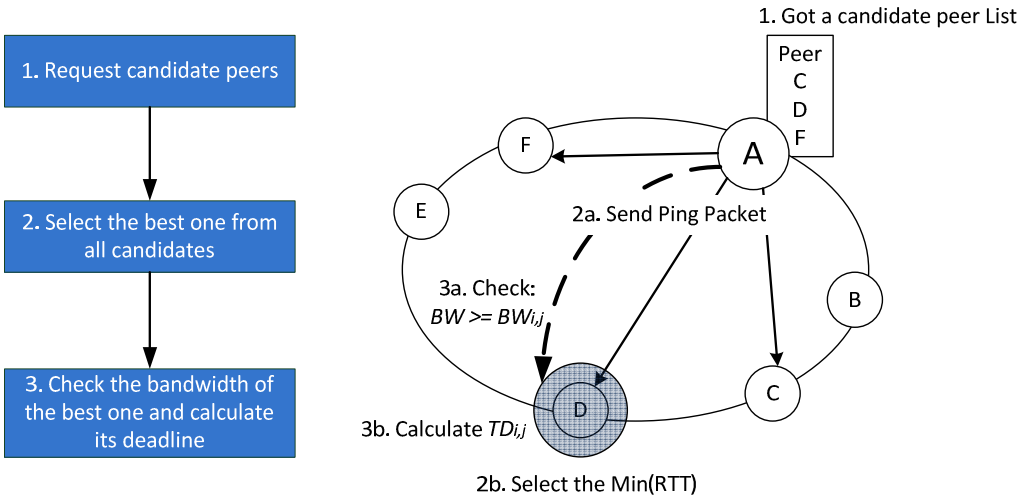


Figure 8　Selection Phase.

Pseudo code 如下圖 9，透過兩個 for 迴圈針對每一個$L_{i,j}$，從其候選節點清單中比較 Ping 封包的 RTT 值，當此節點 k 的 RTT 值為最小又符合$L_{i,j}$所需要的頻寬需求$BW_{i,j}$時，將節點 k 設為最佳節點$B_{i,j}$，並計算相關的傳輸時間$TR_{i,j}$以及最終播放時間$TD_{i,j}$以供排程階段使用。

```
for ( i=0 to i=N-1 )
    for ( j=0 to j=M-1 )
        for( k=1 to C_{i,j} ) {
            if [(RTT_k < RTT_{i,j}* ) && (sizeof(ping)/ RTT_k >= BW_{i,j}) ]
                {   RTT_{i,j}* = RTT_k ;
                    B_{i,j} = k ;
                    TR_{i,j}  = S_{i,j}  / BW_{i,j}
                    TD_{i,j} = D_i - TR_{i,j}
                }
        }
```

Figure 9　The Pseudo Code of Selection Phase.

4.3　Scheduling Phase

在這個部分裡，為了達到在播放時無延遲的目的，就必須在每個區塊 i 的最終播放時間D_i內完成區塊 i 的傳輸；若希望區塊 i 能達到最高的畫質效果，則必須取得最多個區塊 i 的$L_{i,j}$。因此綜合以上兩點，「在最終播放時間D_i內，完成最多個$L_{i,j}$的傳輸」是我們的最終目標。

Components of Scheduling

首先我們要考慮的有三個部分，如圖 10 的 LDS 區域：

1. 最終播放時間(Deadline)

當取得$L_{i,j}$的時間已經超過它的最終播放時間$TD_{i,j}$，此時$L_{i,j}$已經無用處。為了避免收到$L_{i,j}$時超過$TD_{i,j}$，造成不必要的傳輸，我們將「最終播放時間」作為第一考量，進行傳輸順序的安排。

2. 相依性(Dependency)

由於每個階層之間無法獨立完成解碼，存在階層間的相依性問題，若是基礎層不存在，其後續的增強層就沒有用處。因此給予基礎層最高的優先權值，增強層則給予次高的優先權值。

3. 下載頻寬(Downlink bandwidth)

根據(2)以及(3)，在計算$L_{i,j}$的$TD_{i,j}$時，是以$L_{i,j}$所需的基本頻寬$BW_{i,j}$為基礎，因此若是進行傳輸時的頻寬BW小於$BW_{i,j}$，傳輸時間$TR_{i,j}$勢必會增加，因而影響到$TD_{i,j}$。

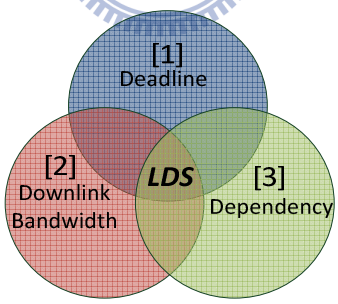


Figure 10　A Conceptual Diagram of LDS Algorithm.

考慮上述因素，分三種狀況討論如表 7，當傳輸時的可用頻寬BW大於$BW_{i,j}$時表示有足夠的傳輸能力，因此不管其優先權值都可進行下載。但當BW小於$BW_{i,j}$時，如果其優先權值大於清單的前一個檔案時，表示目前這個檔案較重要，為滿足相依性需求則必須先取得，因此將清單中的前一個項目刪除，重複檢查此步驟直到BW大於$BW_{i,j}$。但當優先權值小於清單的前一個檔案時，則表示目前的檔案可有可無，為了節省傳輸頻寬而將其放棄下載。

Table 7　Three Cases of LDS.

Case	BW	$P_{i,j}$	Handle
1	$\geq BW_{i,j}$	---	Download *list[k]*
2	$< BW_{i,j}$	$> P_{list[k-1]}$	Drop *list[k-1]* , check again
3	$< BW_{i,j}$	$< P_{list[k-1]}$	Drop *list[k]*

Scheduling Procedure

為了簡化過程，我們將一個完整的串流資料分段處理，每次處理N個區塊(Chunk)的資料。如下圖 11 對於這N個區塊進行排程的步驟如下：

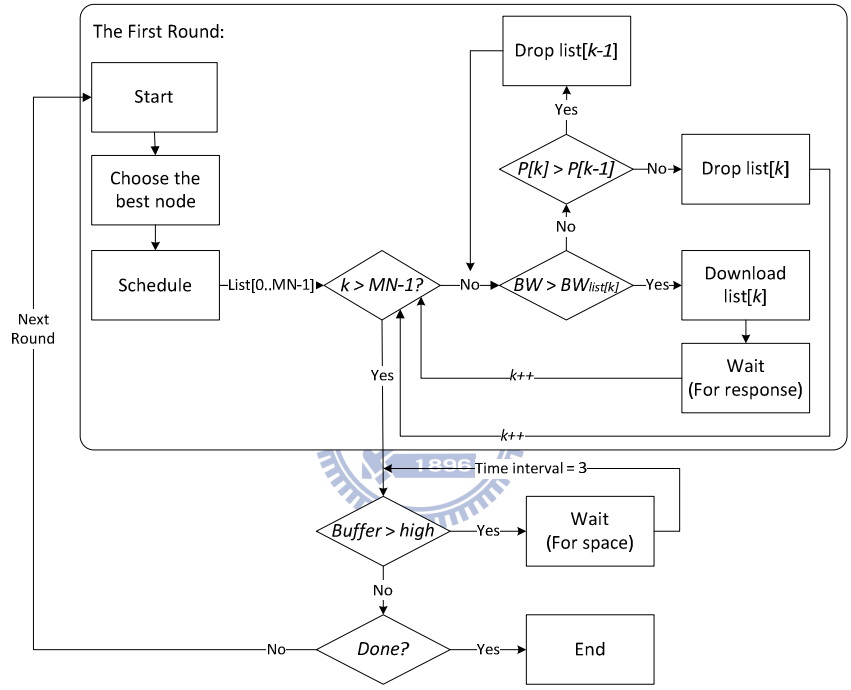


Figure 11　The Flow Chart of LDS.

1. 為了避免取得的階層超過其最終播放時間而失效，因此將所有需要的階層$L_{i,j}$依照其最終播放時間$TD_{i,j}$進行 最早期限優先(Earliest Deadline First，EDF)排程。
2. 若實際傳輸時間超過當初所預估之時間，則用來排程下載順序的最終播放時間將失去準確性，因此為了保證先前所預估的傳輸時間無誤，將依照順序檢測當時的下載頻寬BW是否符合該階層的所需頻寬$BW_{i,j}$？
   i. 若符合就進行傳輸，如表 7 的 Case 1。
   ii. 若不符合但目前要傳輸的階層優先權值大於序列中前一個階層，便停

止前一個階層的傳輸，並重複此步驟，如表 7 的 Case 2。

iii. 若頻寬不符合且要傳輸的階層優先權值也比序列中前一個階層要來的小，則表示這個階層非必要，因此可將其刪除，已完成後續的傳輸。，如表 7 的 Case 3。

根據最早期限優先(EDF)的排程原則，我們可以在期限內完成所需階層的傳輸，避免播放時的延遲。而靠著階層間的優先權值關係，我們則可以盡可能的取得更多的階層來改善畫面品質。

4.4 Example Run

程序流程範例如下圖 12，步驟 1 先將每一回合(Round)的所有物件(Object)根據各自的最終下載時間$TD_{i,j}$排序成 *list[0..MN-1]*；步驟 2 判斷該回合的物件是否都處理完畢，若尚未結束則接著檢查當時的可用頻寬是否符合$L_{i,j}$的最小需求頻寬$BW_{i,j}$，若符合則進行步驟 3.1 開始 *list[k]*的傳輸；若不符合則接著步驟 3.2 進行優先權值的比較，當目前的 *list[k]*權值較高則進行步驟 4.1 中斷 *list[k-1]*的傳輸並重覆步驟 3 檢查可用頻寬；當目前的 *list[k]*權值較低則進行步驟 4.2 取消 *list[k]*的傳輸，待該回合的所有物件皆處理完畢，接著步驟 5 開始下一回合的排程程序。

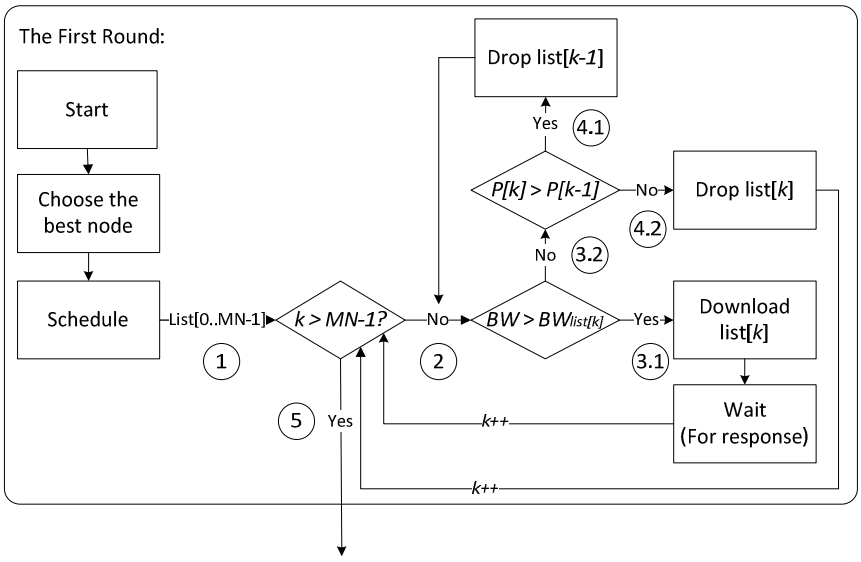


Figure 12 The Example Run of LDS.

5 System Implementation

在這篇論文中，我們建構了一個系統平台來驗證前一章所提出的排程方法。如下圖 13 所示，系統主要分為兩個部分：一是處理串流檔案的前置作業，負責將串流檔案處理成所需要的階層格式，再將這些階層檔案放置到我們所建構的 Chord 網路環境中，讓一節點可利用此 P2P 環境取得所需的檔案，詳細如 5.1 節所述；二則是讓依使用者可以看到串流影像的觀看程序，包含前一章所提及的 LDS 排程演算法，詳細如 5.2 節所述。

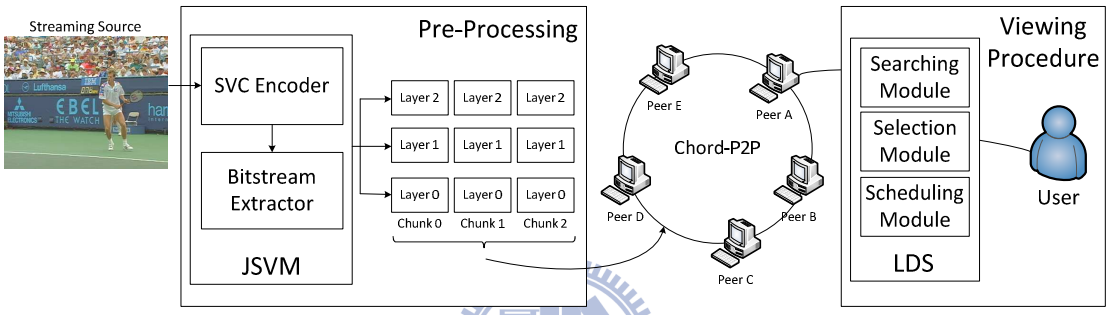


Figure 13 An Overview of System Implementation.

5.1 Pre-Processing

處理串流檔案的前置作業如下圖 14 所示，將一個串流檔案經由 JSVM 6.8.2 所提供的 SVC Encoder 壓縮成 SVC 檔案，再透過 Bitsream Extractor 萃取出各階層的內容，再放置到 Chord-P2P 環境中。

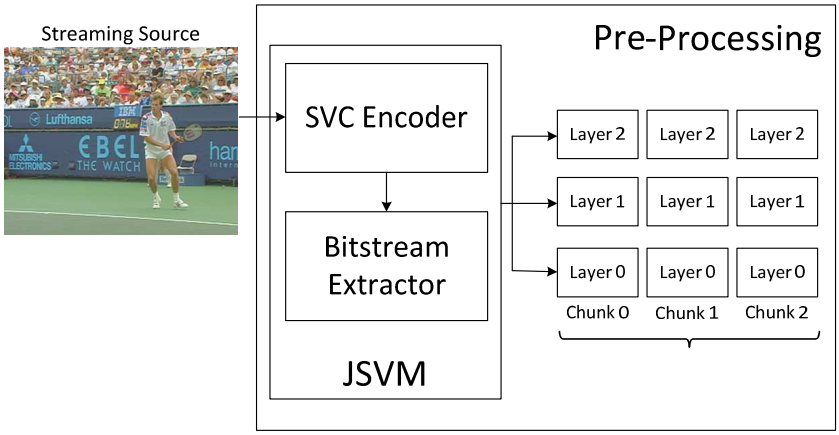


Figure 14 An Overview of Pre-processing.

5.2 LDS Algorithm

在 LDS 排程演算法裡，我們使用 VB.NET 作為程式撰寫之基礎並執行於 Windows 平台上。其架構如下圖 15 所示，分成三個模組：搜尋模組(Searching Module)、選擇模組(Selection Module)、排程模組(Scheduling Module)。其中搜尋模組主要功能為尋找某一特定階層檔案的網路位置，*Hash File Name* 程序將每個$L_{i,j}$檔案的檔名進行雜湊(Hash)後取得 hash_id，*Compare Hash ID* 程序再透過 Chord-P2P 環境，利用雜湊後的 hash_id 進行字串比較，尋找在網路中擁有這個檔案的候選節點群並回傳一候選節點清單(List of Candidate Peers)。

此清單被搜尋模組從中挑選出一個最適合的節點來取得須要的檔案，由下式(5)可知 RTT 與頻寬成反比，在不考慮封包遺失率的情況底下，若與某個節點的 RTT 較小，則表示其頻寬越大。

$$\text{Bandwidth} = \frac{1.3*\text{MTU}}{\text{RTT}*\sqrt{\text{Packet Loss}}} \qquad (5)$$

因此為了求的(5)的頻寬，圖 14 中的 *Send Ping Packet* 程序負責送一 Ping 封包至候選節點清單內的每一個候選節點，由於最適合的節點必須具備最短的 RTT，因此 *Min(RTT)* 程序則利用 Ping 封包的來回時間求出最適合的節點下載$L_{i,j}$。並將此節點資訊儲存於資料結構中，供後續的下載程序使用。

最後的排程程序以資訊清單結構中的所有$L_{i,j}$進行最佳化的排程程序，以達到「在每個 *Chunk i* 的最終播放時間D_i內，完成最多$L_{i,j}$的傳輸」的目地。在後續的步驟裡我們以N個區塊為單位進行每一個回合(Round)。圖 14 中的 *Sort List* 程序先將資訊清單結構裡的所有$L_{i,j}$依照其最終播放時間$D_{i,j}$大小排序成一個下載清單結構；而 *Check Bandwidth*、*Check Priority* 程序則根據表 7 的規則，檢查下載清單結構裡每個$L_{i,j}$的頻寬以及優先權值，判斷是否該進行下載或是取消，以達到最佳的下載順序；最後 *Download File* 程序先產生一執行緒(Thread)再利用 HTTP Protocol 進行$L_{i,j}$檔案的下載。在本地端會有一塊空間用來暫存待播放的影像稱為緩衝區(Buffer)，每完成一個回合的排程後，為了節省網路資源，*Check*

Buffer 程序發現緩衝區的大小超過所設定的門檻值時，表示緩衝區內的資料太多還不急著進行下一回合的排程及下載，等待一段時間後，再檢查一次，若緩衝區的大小低於門檻值，則可進行下一回合的排程程序。

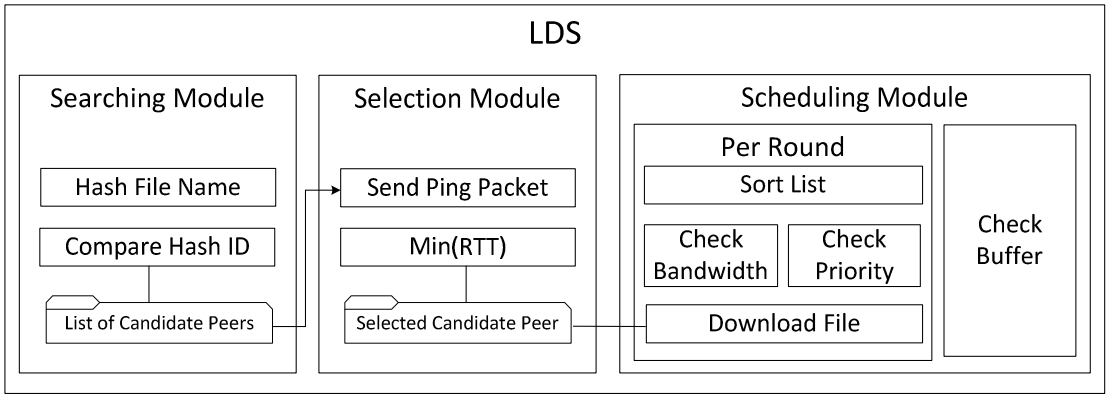


Figure 15　An Overview of LDS Procedure.

6 Experiments and Results

6.1 Test Environment

實驗的環境包含了三個角色 Client、WAN Emulator 以及 Peer A、B，如下圖 16。其中 Client 負責在 P2P 網路中搜尋一段串流視訊來觀看； WAN Emulator 則是用來模擬真實網路環境，藉此消除內部測試網路環境過於單純之問題，除了限制各點之間的點對點頻寬之外，實際測試時並針對封包的延遲時間(RTT/2)、遺失率(Loss Rate)、重複率(Duplication Rate)、錯誤率(Corruption Rate)進行參數之設置；最後 Peer A、B 負責模擬 P2P 網路中大量的節點，並提供串流視訊檔案。

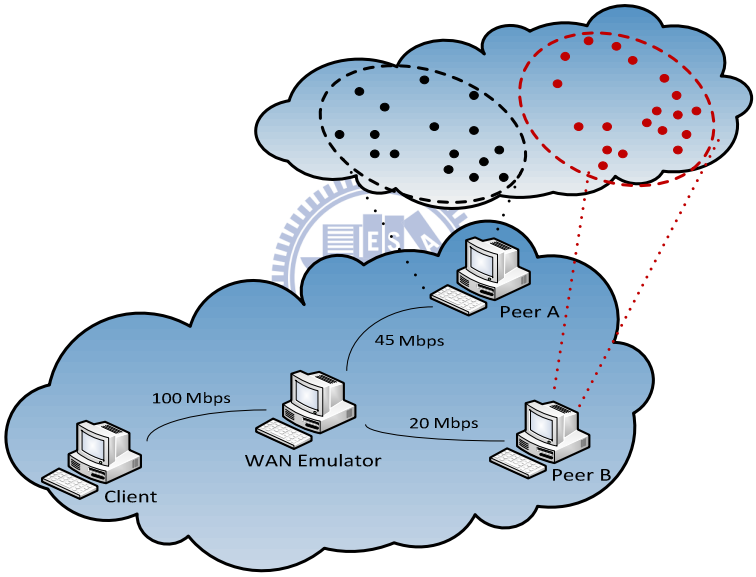


Figure 16 Test Environment.

6.2 Test Cases

將一個十二秒的串流視訊以每一秒的時間間隔分割成十二個不同區塊(chunk)，再將每一個區塊分別用 JSVM 壓縮成三個不同品質的階層檔案(layer)：Baseline @ Level 1.3、Scalable Baseline @ Level 2.1、Scalable High @ Level 2.2。

考慮將不同的區塊個數進行每一回合的 LDS 排程程序，若區塊個數過少則會因下載程序會過於頻繁而造成同時有太多傳輸進行使得傳輸時間增加，反之，若區塊個數過大表示同時有過多的階層檔案進行傳輸，也會使得傳輸時間增加，

從表 8 中可得知當區塊個數為三時所得到的效果為最好，也就是每回合排程三個區塊，因此後續測試的每一回合皆排程三個區塊為單位。

Table 8　The Number of Chunks Per Round.

Number of Chunks Per Round	2	3	4	6
Start-up Latency (sec)	20	14	15	35
Waiting Delay Time (sec)	51	17	41	5
Total Delay Time (sec)	71	31	56	40

6.3　Test Results

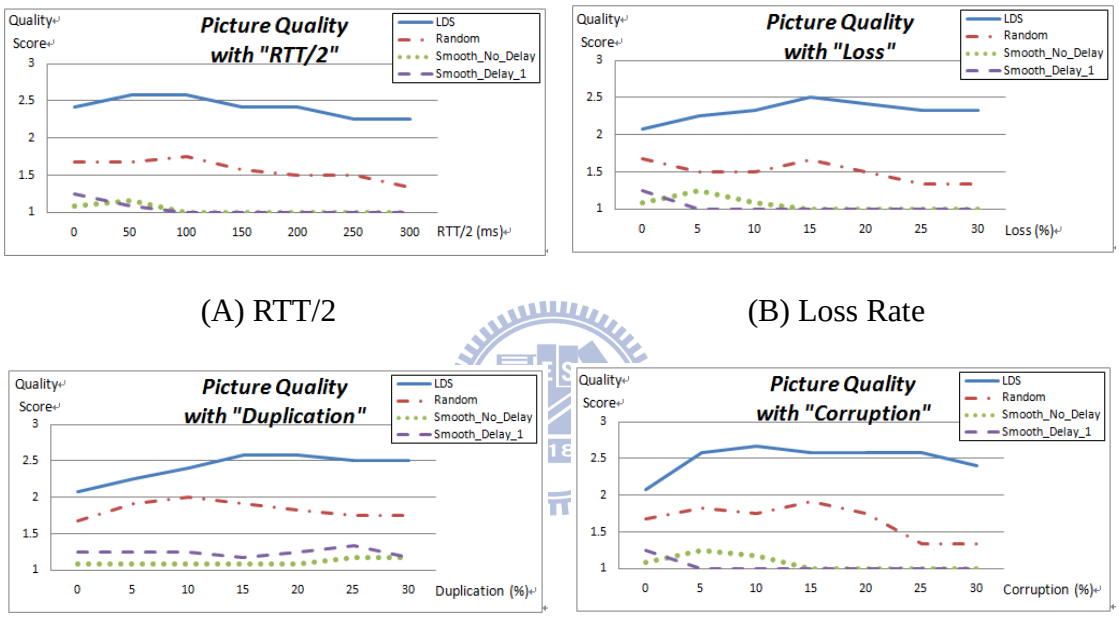
為了符合真實網路環境，我們利用 WAN Emulator 對分別對封包的延遲時間、遺失率、重複率、錯誤率等參數作設置，並針對挑選節點以及排程方法進行測試項目的比較，其中我們觀察以下四種演算法：(1)本論文所提的 *LDS Algorithm*、(2)不做任何處理的 *Random Selection Algorithm*、以及第二章的相關研究中所提及的(3)*Smooth Algorithm without delay* 和(4)*Smooth Algorithm with 1 sec*[5]，以了解畫面品質(Picture Quality)、演算法效能(Performance of Algorithm)、使用者等待時間(User Waiting Time)與起始延遲時間(Startup Latency 數值)的影響。同樣於相關研究中所提的 GaiaSharp[6]系統，由於其方法與內部元件結合較緊密，因此無法分離出單一模組針對挑選節點與排程方法進行以上實驗觀察，為了不違反公平性比較原則，不列入此次評測。

Picture Quality

在畫面品質影響的部分，我們將每個區塊所能觀賞到的階層數量作評分，若在播放時間截止時只能觀賞到基礎層(Base Layer)則分數為 1，以此類推，因此在本實驗中的品質分數最高為 3，最後再將所有區塊的品質分數取平均。品質分數越高表示觀看的畫面品質越好。

如下圖 17 所示，當我們分別將網路環境的延遲時間(A)、遺失率(B)、重複率(C)、錯誤率(D)等參數作設置時，明顯地因為 LDS 演算法能有效的在每個區塊的最終播放時間內取得最多的階層數，因此 LDS 不管在任何時候都是表現最

好的，其畫面品質接近滿分 3；而 Random 是隨機挑選節點以取得檔案，因此挑到 RTT 值大的節點機率較高，造成畫面品質不穩定；最後 Smooth_No_Delay 以及 Smooth_Delay_1 固定從 RTT 值最小的節點取得基礎層，RTT 值次小的節點取得增強層，因此當網路環境越來越差時取得增強層的時間便會增加，使得增強層來不及在最終播放時間之內取得，造成其畫面品質無法提升，僅維持在基本品質的部分。因此 LDS 能有效的改善畫面品質，並改善了 Smooth Algorithm 將近一倍以上的畫面品質。

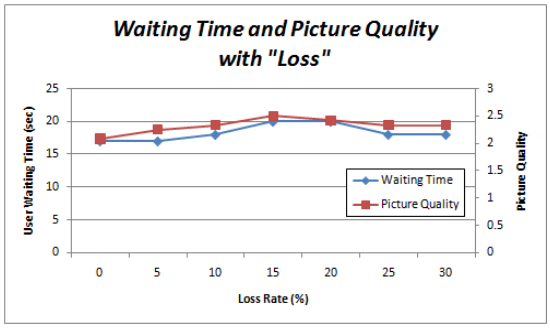


(A) RTT/2　　(B) Loss Rate

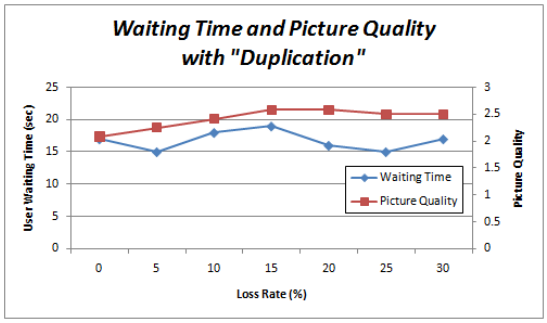
(C) Duplication Rate　　(D) Corruption Rate

Figure 17　Picture Quality.

但觀察圖 17(B)(C)的畫面品質曲線部分，當封包遺失率與重複率逐漸增加時 LDS 所能提供的畫面品質並沒有因此而下降。所以我們對照使用者等待時間的數據圖如下圖 18，同時參考使用者等待時間對畫面品質的影響時，發現這兩條曲線的走向是一致的，也就是說當使用者等待時間增加時，由於緩衝的時間增加使得有充裕的時間取得更多的階層，造成畫面品質隨著遺失率與重複率增加實並無下降。



(A) Loss Rate



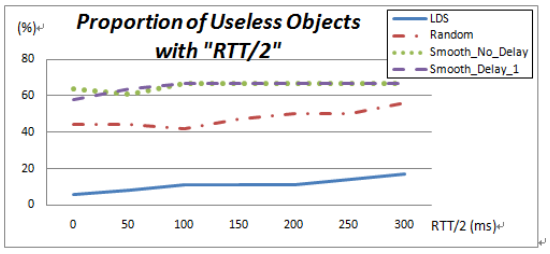
(B) Duplication Rate

Figure 18　Waiting Time and Picture Quality.

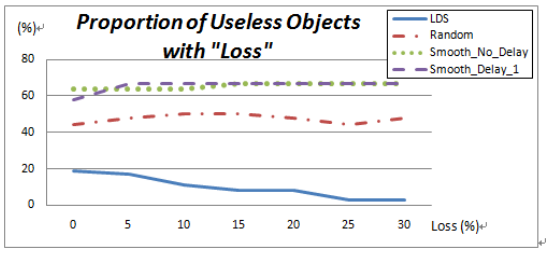
Performance of Algorithm

我們利用「失效下載率」來當作演算法效能的指標。失效下載率指的是下載了多少失效的檔案，當取得某個增強層後因取得的時間已經超過其最終播放時間，則這個增強層便無用處了。當失效下載率越高表示下載越多無用處的檔案，不僅對畫面品質的改善沒有幫助之外，也多增加了不必要的網路流量。

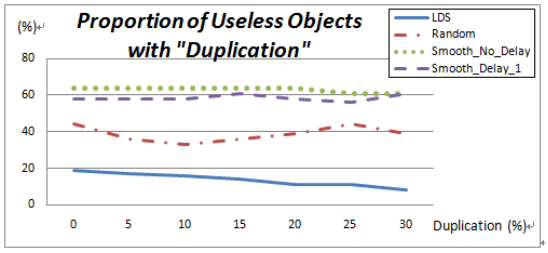
如下圖 19 所示，當我們分別設置延遲時間(A)、遺失率(B)、重複率(C)、錯誤率(D)等參數時，由於 LDS 演算法預先估算了每個階層的傳輸時間，當預測結果顯示某個階層檔案來不及在其最終播放時間之前取得，便提早決定放棄這個可能會是無效的階層，使得 LDS 所取得的檔案幾乎都能完全的增加畫面品質，其失效下載率最低；而 Smooth_No_Delay 以及 Smooth_Delay_1 則是因為增強層皆由較差的節點取得，因此傳輸的時間較長，造成完成下載時已經超過其最後播放時間，使得失效下載率高居不下。因此 LDS 演算法有最高的排程效能，並為 Smooth Algorithm 的三倍左右。

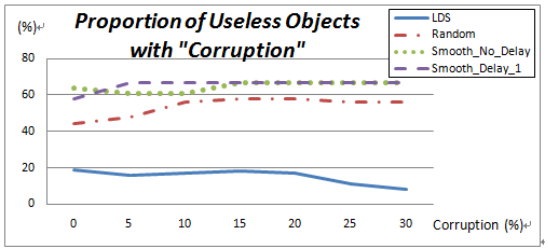


(A) RTT/2



(B) Loss Rate





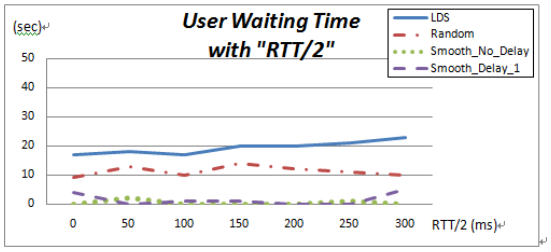
(C) Duplication Rate (D) Corruption Rate

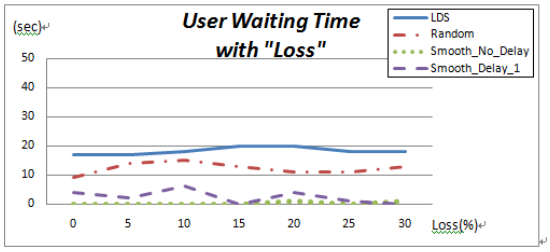
Figure 19 The Performance of Algorithms.

User Waiting Time

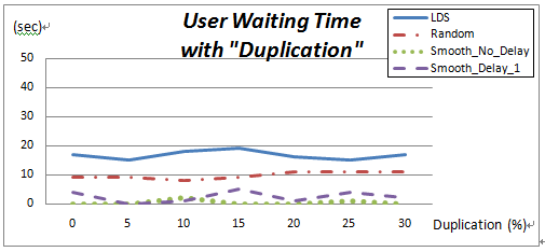
在觀看串流片段的途中，有些區塊在其播放時間之前尚未取得時，必須等待一段緩衝時間等區塊下載完成，才能夠繼續串流片段的觀賞。這些所有等待的緩衝時間累加就為「等待延遲時間」。當等待延遲時間越短則表示該排程演算法越能夠有效率的利用每個區塊的最終播放期限。

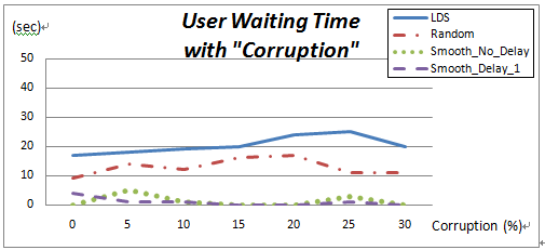
如下圖 20 所示，當我們分別設置延遲時間(A)、遺失率(B)、重複率(C)、錯誤率(D)等參數時，LDS 所需要的等待時間稍長一些；而 Smooth_No_Delay 以及 Smooth_Delay_1 則是因為固定由最佳的節點取得基礎層因此傳輸的時間較快，使得等待時間最短，但也因此造成畫面品質低落以及失效下載率最高。





(A) RTT/2 (B) Loss Rate

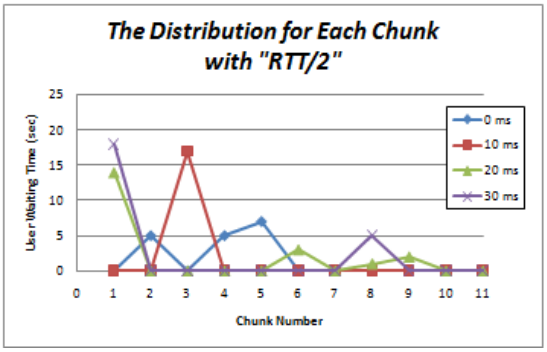




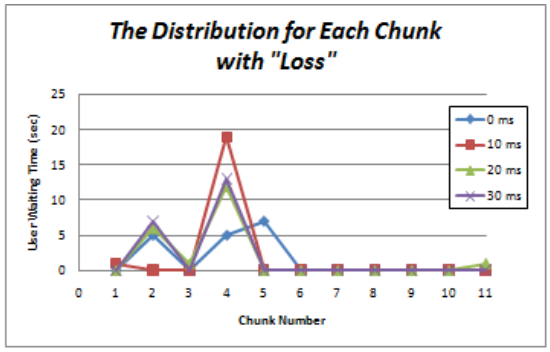
(C) Duplication Rate (D) Corruption Rate

Figure 20 User Waiting Time.

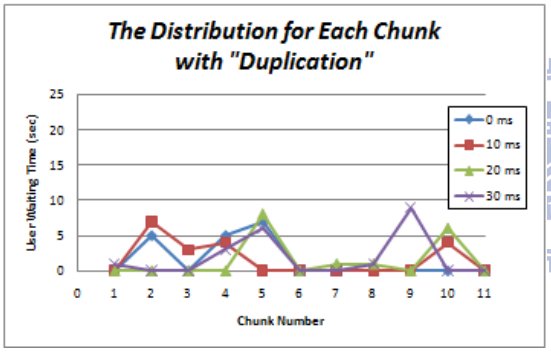
下圖 21 說明了 LDS 在各種其況下各個區塊的等待時間分布圖，當延遲時間(A)增加時除了一開始的區塊會有部分等待時間產生之外，在後面的區塊也產生了一些等待時間。而當遺失率(B)增加時其等待時間都集中在前面的區塊。重覆率(C)與錯誤率(C)增加時等待時間的分布則都很均勻。



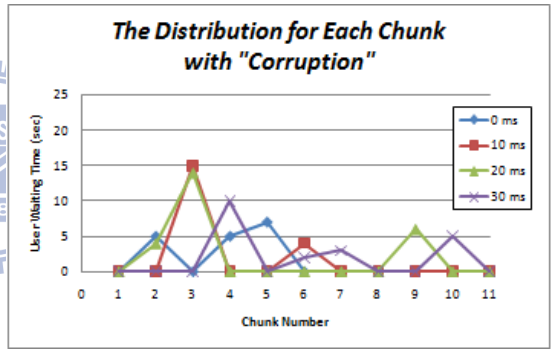
(A) RTT/2



(B) Loss Rate



(C) Duplication Rate



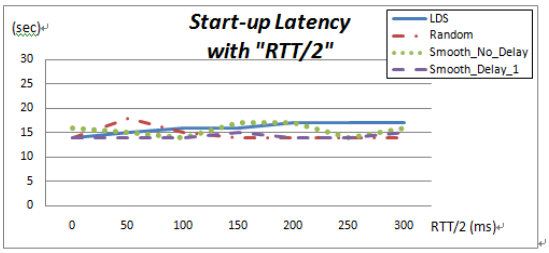
(D) Corruption Rate

Figure 21　The Distribution for Each Chunk.

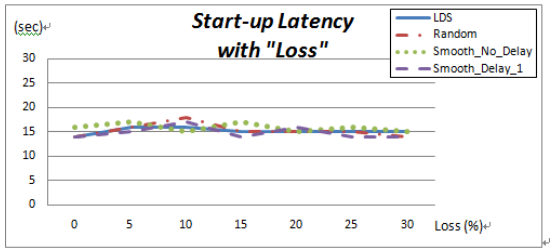
Start-up Latency

當使用者送出觀看串流影像的需求一直到使用者真正看到影像的時間稱作「起始延遲時間」，也就是一開始所等待緩衝的時間。

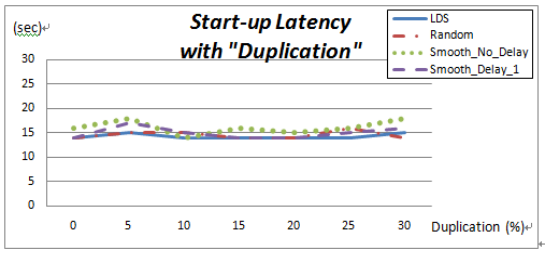
如下圖 22 所示，當我們分別設置延遲時間(A)、遺失率(B)、重複率(C)、錯誤率(D)等參數時，LDS、Random、Smooth_No_Delay 以及 Smooth_Delay_1 的都落在 15 秒左右的平均值裡。



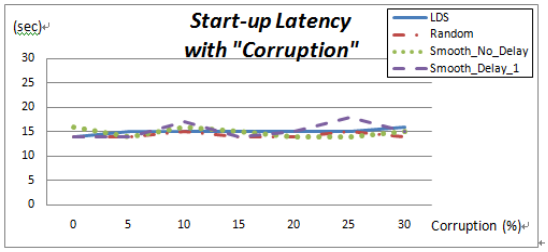
(A) RTT/2



(B) Loss Rate



(C) Duplication Rate



(D) Corruption Rate

Figure 22　Startup Latency.

Picture Quality v.s. User Waiting Time

針對 QoE 的兩項影響因素：畫面品質以及使用者等待時間，從上述結果可得知 LDS 在畫面品質部分有最好的表現但讓使用者等待一段時間，相反的，Smooth Algorithm 卻是讓使用者等待較短的時間但只提供基本的畫面品質。因此我們想知道在網路環境相同的條件下，若讓 Smooth Algorithm 提供較好的畫面品質時，其等待時間是否依舊較短，如下圖 23 所示，一開始 LDS 提供基礎畫質時須讓使用者等待一些時間，但當畫質需求漸升時，反而 Smooth Algorithm 所需要的等待時間越來越長，甚至超過 100 秒，這是因為所有區塊的增強層都是由 RTT 值次小的同一節點取得，使得此節點的負擔太重造成所有增強層的傳輸時間加長。因此可得知 LDS 能在有限的等待時間之內提供使用者最佳的畫面品質。

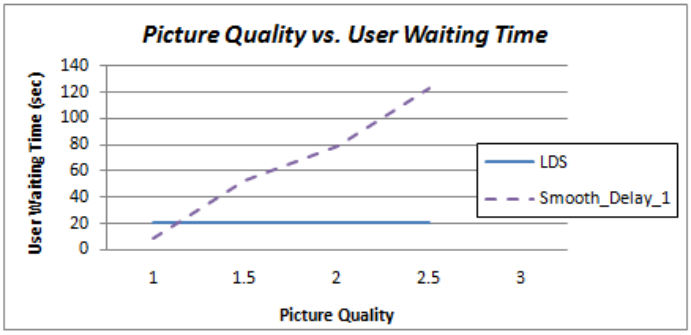


Figure 23　Picture Quality vs. User Waiting Time.

7 Conclusions and Future Works

儘管 P2P 視訊串流技術的應用越來越廣泛，但關於使用者經驗品質(QoE)仍然較少被討論。在本文中，我們考量到 P2P 網路中大量的節點各自擁有不同的使用需求，為了滿足不同使用者的需求，我們利用可調式視訊編碼的特性，讓不同的使用者可依需求觀看不同畫質的視訊影像。

Layer Deadline Scheduling 演算法依每個階層區塊的最終播放時間進行排程，再利用不同階層的優先權值以及當時的可用頻寬來進行下載順序的調整，如此便可確保在使用者等待的時間不至於過程的情況下可獲得最佳的畫面品質。

要改善使用者經驗品質可從畫面品質以及等待時間這兩個影響因素著手，從表 9 的實驗結果得知我們所提出的演算法 LDS 能提供最佳的畫面品質以及演算法效能，但卻必須付出讓使用者等待的代價。雖然 Smooth Algorithm 的使用者等待時間較短，但上圖 23 顯示當 Smooth Algorithm 所提供的畫面品質越好時所需的等待時間也越大。

Table 9 The Comparison of Results.

	LDS Algorithm	Random Selection	Smooth Algorithm
Picture Quality	Best	Worse	Worst
Performance of Algorithm	Best	Worse	Worst
User Waiting Time	Longer	Longer	Shorter

綜合以上比較可得知 Layer Deadline Scheduling 演算法能有效的改善 QoE，但事實上，若不考慮畫質改善的部分多重描述編碼(MDC)是較適合用於 P2P 串流服務，因為 MDC 的每個描述子(Description)間各自獨立，不存在解碼時彼此依賴的問題，當某個描述子來不及傳輸時並不會有解碼失敗的情況發生。雖然 MDC 適用於 P2P 串流服務，但僅於學術上的討論並無實際的應用，因此考慮 MDC 並應用在 P2P 串流系統服務是我們下一步欲實現的功能。

Reference

[1] G. Harman, "The Intrinsic Quality of Experience," *Philosophical Perspectives,* vol. 4, Action Theory and Philosophy of Mind (1990), pp. 31-52, 1990

[2] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1129, Sep. 2007

[3] X. Hei, Y. Liu, and K.W. Ross, "IPTV over P2P Streaming Networks: The Mesh-Pull Approach," *IEEE Communications Magazine*, vol. 46, pp.86–92, 2008

[4] M. Mushtaq, T. Ahmed, "Hybrid Overlay Networks Management for Real-Time Multimedia Streaming over P2P Networks," *Proceedings of the 10th IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services: Real-Time Mobile Multimedia Services*, vol. 4787, pp.1-13, 2008

[5] M. Mushtaq, T. Ahmed, "Smooth Video Delivery for SVC Based Media Streaming over P2P Networks," *5th IEEE Consumer Communications and Networking Conference*, Las Vegas, NV, pp. 447–451, 2008

[6] T.C. Lee, P.C. Liu, W.L. Shyu, C.Y. Wu, "Live Video Streaming Using P2P and SVC," *Proceedings of the 11th IFIP/IEEE international conference on Management of Multimedia and Mobile Networks and Services: Management of Converged Multimedia Networks and Services*, vol. 5274, pp. 104-113, 2008

[7] "JSVM software", http://ube.ege.edu.tr/~boztok/JSVM/SoftwareManual.pdf

[8] W.P.K. Yiu, X. Jin, S.H.G. Chan, "Challenges and Approaches in Large-Scale P2P Media Streaming," *IEEE Multimedia* , vol. 14, no. 2, pp. 50-59, 2007

[9] X. Zhang, J. Liu, B. Li, and T.S.P. Yum, "CoolStreaming/DONet: A Data-Driven Overlay Network for Efficient Live Media Streaming," In *Proceedings of IEEE INFOCOM*, vol. 3, pp. 13-17, 2005

[10] "PPLive", http:// www.pptv.com/

[11] "PPStream", http://www.ppstream.com/

[12] "TVAnts", http://www.tvants.com/introduction/superiority.html

[13] "Joost", http://www.joost.com/

[14] E. Setton, J. Noh and B. Girod, "Low Latency Video Streaming Over Peer-To-Peer Networks," *Multimedia and Expo (ICME), IEEE International Conference*, pp. 569-572, July 2006

[15] C. Wu, B. Li, "Optimal Peer Selection for Minimum-Delay Peer-to-Peer Streaming with Rateless Codes," *Proceedings of the ACM workshop on Advances in peer-to-peer multimedia streaming*, pp. 69-78, 2005

[16] A. Vlavianos, M. Iliofotou, M. Faloutsos, "BiToS: Enhancing BitTorrent for Supporting Streaming Applications," *INFOCOM 2006, Barcelona, Spain*, pp. 1-6, April 2006

[17] J.F. Paris, P. Shah, "Peer-to-Peer Multimedia Streaming Using BitTorrent," *IPCCC 2007, New Orleans, LA*, pp. 340-347, 2007