淡江大學資訊工程學系碩士在職專班

碩士論文

指導教授：黃連進　博士

自適式影像編碼 HLS 系統

Adaptive Transcoding HTTP Live Streaming System

研究生：呂建億　撰

中華民國 102 年 6 月

致謝

本論文能順利完成，感謝指導教授黃連進博士的支持，在碩士一年級平時已被工作和上課擠滿時間，能於第二年順利完成，在於老師給予最大空間的自由，方能在有限時間內完成此研究論文，在本人遇到瓶頸與困難時，給予最大幫助。

需要感謝三位好朋友的大力幫助，王玨、李柏漢與李子豪，提供很多過來人的經驗與方向，方能順利完成此研究，感謝碩士同學孫敏鎬、林高慶與何信凱，一起討論一起分擔壓力，也感謝公司同事給予幫助，分擔工作上的壓力與事情，讓本人能有多餘時間專心於學業上各種事務。最後感謝我的知已范維真，於研究後期焦慮症嚴重發作時，給予心靈上的平靜，排解各種壓力後，安心完成此論文。

要感謝的人太多，謝謝老天讓本人身邊有各種朋友與長輩，提供各種協助，特此致謝。

呂建億　謹於淡江大學資訊工程研究所

102 年 6 月

論文名稱： 自適式影像編碼 HLS 系統 頁數：58

校系(所)組別：淡江大學 資訊工程學系 碩士在職專班

畢業時間及提要別：101 學年度 第 2 學期 碩士 學位論文提要

研究生：呂建億 指導教授：黃連進 博士

論文提要內容：

現今流行的網路串流，Apple HTTP Live Streaming，其缺點無法隨著客戶端與伺服端的頻寬變化，提供相對應的位元率，導致使用者觀看品質不流暢。所以此篇論文是希望能解決以上遇到問題，能利用動態編碼和頻寬管理，達到即時影像可依據頻寬變化，而會改變輸出的影像品質，先以求達到流暢播放為目的，而後求保持最大頻寬使用量下，提供最高的播放品質。

關鍵字：HTTP Live Streaming；行動串流；頻寬偵測；可適性串流；WebVTT

表單編號：ATRX-Q03-001-FM030-01

Title of Thesis: Adaptive Transcoding HTTP Live Streaming System

Total Pages: 58

Key word: HTTP Live Streaming; Mobile Streaming; Bandwidth Detection; Adaptive HTTP Streaming; WebVTT

Name of Institute: Executive Master's Program, Department of Computer Science and Information Engineering, Tamkang University

Graduate Date: 06/2013

Degree conferred: Master

Name of Student: Chien-Yi Lu 呂建億

Advisor: Dr. Lain-Jinn Hwang 黃連進 博士

Abstract:

The most popular Internet live streaming technique, Apple HTTP Live Stream, which causes that users suffer from the consistency in bitrate when the bandwidth between the user and the server varies. Thus, this article solves the issue mentioned previously. The method we conduct makes use of adaptive transcoding technique and bandwidth management approach in order to achieve the goad that the video stream quality varies in according to the bandwidth change. We have the stream fluency the major consideration while the maximum bandwidth utilization the minor to assure the highest video quality.

表單編號：ATRX-Q03-001-FM031-01

目錄

圖目錄

表目錄

第一章　緒論

1.1　研究動機

現在網路的發達，網際網路上傳和下載頻寬速度越來越快，不管是家用網路或是行動網路，所以透過網路為媒介的多媒體服務逐漸多樣化，其中網路串流更是常見，從早期的網路電台，到現在可提供高畫質的網路視訊串流，使用者已經不是單純只是利用設備瀏覽網頁。根據研究顯示，世界網際網路流量大增，超過一半流量以上都被網路串流佔用 [1] [2]。現今智慧型手機和平板在市場上數量逐漸增加，很多各種雲端服務，都會透過手機和平板來實現，所以網路串流可提供的平台更加多元化，雲端服務已經從早期在室內定點使用有線網路，演變成隨時隨地可以在室外各種地方上網 [3]。智慧型手機和平板螢幕尺寸越來越大和提供更高的解析度，如果視訊畫質和解析度沒有相對提高，反而看到的觀看畫質會更差 [3]。台灣手機上網流量在近幾年大增，最新行動網路頻寬測試顯示，台灣手機網路相當不穩定，在各個地方上網速度有所差異，例如在移動的捷運上速度為最慢，而且手機上網速度也會受上網時間點影響，可區分為睡眠時間到上班前平均手機上網速度為佳，上班時間到清晨一點為差，儘管現在手機網路已經提升至 LTE 高速網路，但台灣還未完成建置，台灣人民無法享受到更高速的網路頻寬，在台灣行動網路建置還不完善時，如果人們利用如果不穩定網路上網瀏覽網頁或是觀看網路串流，其使用體驗將會相當差 [4]。在室外使用手機上網頻寬變化沒有像室內接上網路線使用一樣穩定，會依據當地訊號的強弱、基地台同時使用人數的變化、當地提供的網路頻寬和移動中所產生的頻寬變化等，各種方面都會影響到播放線上串流地流暢度，在頻寬不穩定的網路環境下，如何讓使用者在畫質和流暢度取得平衡，得到最佳的觀看品質，是很重要可提出來討論的方向 [5] [6]。

傳統提供網路串流，一定需要在電腦安裝同樣的伺服端和客戶端的程式，伺服端壓縮固定位元率 (影像品質) 且相同的影像編碼格式，透過 TCP 或 UDP 或 RTP/RTCP 等傳輸協定，傳送影像和聲音給客戶端，客戶端再對接收到的視訊串流進行解碼且播放。由於使用固定位元率編碼且是預先編碼好，所以如果在頻寬足夠的情況下，可以輕易提供高畫質的影片，但是當現有頻寬無法播放此位元率影片時，

就容易發生一直在緩衝影片的狀態。現在是行動網路盛行的時代，人們生活節奏也講求快速的時期，需要能快速獲得所需要的資訊或即時性的服務，不應該讓使用者花費時間等待影片的下載，所以本篇論文在於希望能解決此問題，使用者可以不用事先編碼好支援播放得影片格式，即可提供使用者即時觀看且高畫質的線上串流影片。

現在網路平台眾多，傳統觀看線上串流方法，使用者一定要安裝可提供播放此線上串流服務相對應的程式，方可播放且觀看影片，需要由使用者去思考播放程式的選擇，不是每個使用者都清楚知道該如何找到相對應的播放程式，應該提供使用者更簡便的方式收看線上串流，不應該由使用者去煩惱該安裝什麼播放程式，該選擇那種播放格式。如果客戶端只需要打開瀏覽器即可使用各種服務，可提供使用者更方便的服務，簡化使用者操作上的問題，即可提升使用者觀看體驗。

1.2 研究架構

本研究主要著眼於線上串流如何動態調節編碼，現在標準的網路串流在畫質變化上，提供二種選項，一是提供相同的畫質，所以所需頻寬也是固定的，如果頻寬不足，會導致串流一直停頓，二是串流可能提供多種畫質，由使用者選擇適合自己網路頻寬的畫質，但如果使用者頻寬還是低於所提供最低畫質，還是無法順利播放。如何使得依據使用者現有頻寬變化，自動調整影片位元率且限制可調整位元率的參數，用以達到使用者最佳的播放體驗。同時實作其線上串流平台，提供可實際運作的播放平台，方可清楚了解使用者觀看體驗，從而由實作中隨時修改設計，達到最佳的播放平台。

第一章主要在說明相關文獻回顧，且說明為何此論文的研究動機和重要性，且訂定此論文的研究目標。

第二章將會說明本篇論文使用到所有技術和理論，方可了解後面章節實作時，所可能遇到得問題和為何需要使用此技術。

第三章開始說明本篇論文會遇到什麼問題，在此章節將會提出各種理論和該如何利用各種設計出得模型，解決所會遇到的問題。

第四章會提供詳細的實作流程和所會使用的工具或軟體，該如何實現本篇論文

的理論，且透過測試數據和圖表，用以驗証提出的解決方案是否正確和解決程度。

第五章最後提出實作完後的結論，本篇論文獲得什麼改進，且訂定未來可繼續完成的目標。

第二章　背景研究

現在網路的發達，網路上傳和下載速度越來越快，所以很多透過網路的雲端服務越來越多樣化，其中網路線上串流更是常見，從早期的網路電台，到現在可提供高畫質的網路視訊串流，網路串流已經佔用網際網路流量頻寬五成以上 [2]。而現在智慧型手機和平板的盛行，很多各種雲端服務，雲端服務已經從早期在室內使用定點使用網路，演變成隨時隨地可以在各種地方上網，許多服務都透過手機和平板來實現，網路串流可提供的服務更加多元化。

2.1　串流種類

現今各種網路視訊服務正在急速增加，主要可分為兩種，隨選視訊 (Video on demand) [7] 和即時串流 (Live Streaming) [8] [9]。

2.1.1　隨選視訊

隨選視訊是一套可以讓使用者透過網路選擇自己想要看的視訊 (Video) 內容的系統。使用者選定內容後，隨選視訊系統可以用串流媒體的方式進行即時播放，也可以將內容完全下載後再進行播放。系統的播放模式取決於系統及營運上的需求規劃設計，包括收費機制、內容版權、播放品質、機房系統、傳輸系統、收視端的機上盒等。所有的下載型隨選視訊以及一些串流型的隨選視訊都可以讓使用者以過去操作錄放影機 (VCR) 的方式來使用，包括暫停 (pause)、快轉 (fast forward)、快速回帶 (fast rewind)、慢速播放 (slow forward)、慢速重播 (slow rewind)、跳到前一個/下一個鏡頭畫面等。若是要以串流傳送的方式來實現這些播放操控功能，則遠端機房內的伺服器的工作負荷就會大增，同時也可能需要更大的網路頻寬才行 [7]。

其為線上串流服務之一，可提供使用者經由網路，自由選擇播放影片，其影片來源都是已錄製好的影片，使用者可以選定內容播放或選擇先完全下載之後再觀看，使用方法像是傳統觀看 VCR 的方式，可提供使用者隨時暫停，快轉或是跳轉等操作方式，且支援多使用者使用，在同一個家庭內，每個人都可各自觀看喜歡的影片或段落，現今市面上已有很多相關服務可使用，例如 Youtube，PPStream

等很多服務，而補習班的函授課程，可提供學生線上觀看教學影片，也可在觀看時間內，隨意跳轉影片播放片段。隨選視訊由於路由 (Routing)，又可分為 Unicast Routing 和 Multicast Routing。Unicast Routing 其為單點傳播路由，所以只針對單一使用者，提供其隨選視訊功能，隨選卡拉 OK 即為其分類，且可提供影片播放有快轉和後退功能等。Multicast Routing 其為群播型路由，其利用 Multicast 達到群體播放功能，主要兩個常見服務為 Near Video On Demand(NVOD) 和 Live-TV 等。NVOD 為一種分時式群播型隨選視訊，其特點在於可針對各種不同客群，播放同一種影片且設定不同播放時間。如果某客群願意付高額費用，即可達到不用等待即可馬上觀看即時串流。如果某客群不願付高額費用，會讓使用者等待數分鐘才可接收到即時串流，中間等待時間，串流提供商可在其中間插入廣告等 [10]。

隨選視訊常用於影片分享網站，目前最大的影片分享網站當屬 Youtube。Youtube 成立於 2005 月 2 月，於 2006 年 11 月被美國網路大亨 Google 所併購，其因為提供相當多樣化的影片線上串流，使用者可以將影片分享上傳到 Youtube，由 Youtube 伺服器自動幫使用者壓縮轉檔成可支援格式，讓使用者可以利用瀏覽器且使用 Adobe 公司的 Flash 或 HTML5；播放 Youtube 上所有使用者上傳的各種影片。早期 Youtube 只提供 H.263 的影片播放格式，所以只能提供 320x240 的解析度和單聲道的輸出音質，而後來逐步提高所支援的解析度，而現今已可提供 H.264 的影片播放格式，如果使用者網路環境和電腦終端設備夠好，即可播放高畫質 1920x1080 解析度或更好的線上串流影片。

2.1.2 即時串流

即時串流是一套提供使用者在網路觀看即時視訊，由電視台或是網路電台所提供，用於即時播放電視節目和即時音樂。電視台即時轉播多為無壓縮影像輸出或低壓縮率輸出，才可達到即時轉播影像，但佔用頻寬大，所以多用於擁有固定頻寬的環境下，例如寬頻電視網路頻寬皆擁有固定頻寬，可以確保影像可流暢傳輸給使用者 [9] [11]。而網際網路上即時串流，為了確保流暢播放，會經由影像壓縮且把畫質和解析度調低，用以減少影像傳輸量，達到可在有限頻寬下流暢播放視訊。有名的即時串流網站為 Justin.tv 和 USTREAM，通過網路進行個人線上音樂與視頻廣播的平臺，其特色為可提供使用者直接經由行動裝置中的錄影程式，即

時錄影和壓縮傳輸至網站，其他使用者可安裝 Flash 播放程式在瀏覽器上觀看即時串流視訊。

2.2 串流技術

2.2.1 傳統串流技術

網路視訊串流以技術來分，可分為 push-based 和 pull-based [12]。在 push-based 架構下，當客戶端跟伺服端請求建立 socket 連線後，伺服端會一直傳輸影像封包至客戶端，經由使用 UDP 傳輸串流，用以達到連續不斷的串流傳輸，例如最常見的結合 Real time protocol (RTP) 的 Real Time Streaming Protocol(RTSP) [13] 和 Adobe 公司的 Real time messaging protocol(RTMP)。

即時串流協定 (Real Time Streaming Protocol，RTSP) 是用來控制聲音或影像的多媒體串流協定，並允許同時多個串流需求控制，傳輸時所用的網路通訊協定並不在其定義的範圍內。伺服器端可以自行選擇使用 TCP 或 UDP 來傳送串流內容，它的語法和運作與 HTTP 1.1 類似，但並不特別強調時間同步，所以比較能容忍網路延遲。允許同時多個串流需求控制 (Multicast)，除了可以降低伺服器端的網路用量，更進而支援多方視訊會議 (Video Conference)。

RTP 由 IETF(Internet Engineering Task Force) 的影音傳輸聯盟所製定，其為 IP 網路上點對點 (Peer to Peer) 的即時影音串流傳輸標準，原先設計是希望可用於 Multicast 的影音播放，在 Unicast 的應用相當廣泛，RTP 常會配合 RTSP 應用於線上串流。RTP 是建構於 UDP 上的協定，當傳送影音資料時，會封裝於 RTP 的封包中；由於封裝於 UDP 的切割封包中，所以無法確保影音資料的正確性，可能在網路環境差的地方，容易發生影音損失等，會影響到影音的播放品質，但好處是傳輸速度快，可快速傳輸高畫質的影片。

在 pull-based 架構下，當客戶端跟伺服端請建立 socket 連線後，只有客戶端跟伺服端請求視訊串流時，伺服端才傳送串流，其使用 TCP 協定傳輸串流，可有效減少伺服端系統負荷，且由客戶端單向決定是否還需要繼續傳輸串流，可減少頻寬不必要浪費。早期如果需要經由 HTTP 在網頁上播放影片，可利用在伺服端即時解碼影片成一張張圖片，而瀏覽器可在網頁中利用 JavaScript 語言，一

直讀取解碼出的圖片檔，即可產生簡易的影片播放功能。但受 HTTP 協定限制，可支援解析度差且無法流暢播放圖片，只適用於簡易播放動畫。HTTP 自適式串流下載 (HTTP Adaptive Streaming) 即為 pull-based 架構的一種技術架構且解決以上問題，常見有 Adobe Flash HTTP Dynamic Streaming(Adobe HDS) [14]、Microsoft Smooth Streaming(MS Smooth) [15]、Apple HTTP Live Streaming(Apple HLS) [16] 和 Dynamic adaptive streaming over HTTP(DASH) [17]。

2.2.1.1 RTSP/RTP 的串流方案

RTSP 是一種傳統的串流控制協定，其具有狀態性的特點意味著從一個客戶端開始連線至伺服端一直到連線中斷的整個過程，伺服端會一直監聽客戶端的狀態。客戶端通過 RTSP 協定向伺服器傳達控制命令，如播放、暫停或中斷等。RTP/RTCP(Real Time Transfer Control Protocol) 是點對點基於多點傳送的應用層協定。其中 RTP 用於數據傳輸，RTCP 用於統計、管理和控制 RTP 傳輸，兩者協同工作，能夠顯著提高網路即時串流的傳輸效率。

基於此架構的串流技術方案，伺服端和客戶端之間建立連線之後，伺服器開始持續不斷地發送媒體封包，媒體封包採用 RTP 進行封裝，客戶端控制信息通過 RTSP 信息包以 UDP 或 TCP 的方式傳送。另外類似的串流協定還包括了 Adobe 的 RTMP(Real Time Messaging Protocol) 以及 Real 公司的 RTSP over RDT(Real Data Transport Protocol)，在此不對這些串流媒體協定逐一進行介紹。

2.2.1.2 HTTP 漸進式下載

HTTP 漸進式下載技術與有狀態的 RTSP/RTP 技術相比，採用了無狀態的 HTTP 協定。當 HTTP 客戶端向伺服端請求連線串流時，伺服端將請求的串流傳輸給客戶端，但是伺服端並不會記錄客戶端的狀態，每次 HTTP 請求都是一個一次性獨立的連線。漸進式下載的功能目前主流的播放器皆支援，如 Adobe 的 Flash、微軟的 Silverlight 以及 Windows Media Player。所謂的漸進式下載，即播放器可以在整個媒體文件被下載完成之前即可開始媒體的播放，客戶端及伺服端如果皆支援 HTTP1.1，電腦終端設備還可從未下載完成的部分中任意選取一個時間點開始播放。目前，主流的視訊網站皆採用了 HTTP 漸進式下載的方式來實現網路串流，如 Youtube、優酷網、土豆網等等。

2.2.1.3 RTSP/RTP 和 HTTP 漸進式下載差異

作為最簡單和原始的網路串流解決方案，HTTP 漸進式下載尤其顯著的優點在於它僅需要維護一個標準的 Web 伺服器，其安裝和維護的工作量和複雜性比起專門的串流伺服器來說要簡單和容易得多。

然而其缺點和不足也很明顯。首先是頻寬容易浪費。當一個使用者在開始下載觀看一個內容之後選擇停止觀看，那麼已經下載完成的內容則是對頻寬資源的一種浪費。其次基於 HTTP 的漸進式下載僅僅適用於隨選視訊內容，而不支援即時串流內容。最後此方式缺乏靈活的連線控制功能和智能的流量調節機制。

而基於 RTSP/RTP 的串流媒體系統專門針對大規模串流媒體即時串流和隨選視訊等應用而設計，需要專門的串流媒體伺服器支援，與 HTTP 漸進下載相比主要具有如下優勢。

- 串流播放的即時性。與漸進式下載客戶端需要先緩衝一定數量串流資料才能開始播放不同，基於 RTSP/RTP 的串流媒體客戶端幾乎在接收到第一幀媒體數據的同時就可以啟動播放。
- 支援進度條搜尋、快進、快退等高級 VCR 控制功能。
- 平滑、流暢的音視頻播放體驗。在基於 RTSP 的串流媒體傳輸期間，客戶端與伺服器之間始終保持連線，伺服器能夠對來自客戶端的反饋信息動態做出回應。當因網絡擁塞等原因導致可用頻寬不足時，伺服器可通過適當降低幀率等方式來智能調整發送速率。
- 支援大規模使用者連線。普通的 Web 伺服器主要針對大量小的 HTML 文件下載而進行優化，在傳輸大容量媒體文件方面缺少性能優勢。而專業的串流媒體伺服器在大容量媒體文件硬碟讀取、緩衝區和網路傳輸等方面進行了優化，可支援大量使用者連入。
- 內容版權保護。在漸進下載模式中，下載後的文件緩衝區在客戶端硬碟的臨時目錄中，使用者可將其拷貝至其他位置供以後再次播放。而在基於 RTSP/RTP 的串流媒體系統中，客戶端只在記憶體中維持一個較小的解碼緩衝區，播放後的媒體數據隨時清除，使用者不容易截取和拷貝。此外還可利用 DRM 等版權保護系統進行加密處理。

儘管如此，基於 RTSP/RTP 的串流媒體系統在實際的架設中仍然遇到了不少問題，主要差異在：

- 與 Web 伺服器相比，串流媒體伺服器的安裝、配置和維護都較為複雜，特別是對於已經建有 CDN(內容傳遞網路) 等基礎設施的運營商來說，重新安裝配置支援 RTSP/RTP 的串流媒體伺服器工作量很大；
- RTSP/RTP 協定層的邏輯實現較為複雜，與 HTTP 相比支援 RTSP/RTP 的客戶端軟硬體實現難度較大，特別是對於嵌入式終端設備來說；
- RTSP 協定使用的通訊埠號 (554) 可能被部分使用者網路中的防火牆和 NAT 等封鎖，導致無法使用。雖然有些串流媒體伺服器可通過隧道方式 (Tunnel) 將 RTSP 配置在 HTTP 的 80 端口上使用，但實際建置起來並不是特別方便。

2.2.2 自適式串流技術 (HTTP Adaptive Streaming)

2.2.1中我們談到了基於 RTSP/RTP 的串流媒體技術以及基於 HTTP 的漸進式下載，但是我們可以清楚看到兩種方案都存在著各自的缺點。

HTTP Adaptive Streaming(以下簡稱 HAS) 技術結合了傳統的串流技術和 HTTP 漸進式下載播放的特點，以 HTTP 的方式向使用者傳送媒體內容，該技術的採用可以大大提升使用者的媒體播放體驗，同時該技術降低了伺服器的技術複雜度。基於 HTTP 的傳送方式提升了媒體內容在網路終端設備中的穿透能力，該技術目前已成為視訊串流發展的趨勢。

HAS 技術融合了傳統 RTSP/RTP 串流媒體技術以及基於 HTTP 漸進式下載技術的優點，具有高效、可擴展以及相容性強的特點。

HAS 技術是一種混合的媒體傳遞方式，給使用者的體驗是串流的方式，但是實際上與 HTTP 漸進式下載方式一樣採用 HTTP 協定完成了內容的下載傳遞，但這些媒體內容都被切割成了一系列的媒體分塊進行傳輸，其為 pull-based 傳輸架構，只有客戶端請求下一段媒體分塊時，才進行下一段傳輸。HAS 技術的一個關鍵就是視訊的切片 (Slice)，每個切片的時間長度相同，一般為 2 至 10 秒。在視訊編碼層，這意味著每個切片都由若干個完整的視頻 GOP 組成 (每個分塊都有一個關鍵 I 幀)，以此保證每個分塊都與過去及將來的媒體切片無關聯。

媒體切片儲存在 HTTP Web 伺服器中，客戶端以線性的方式向 Web 伺服器請

求串流切片，並以 HTTP 方式進行媒體切片的下載，當媒體切片傳輸至客戶端時，客戶端按照順序播放這一系列串流切片。因為這些串流切片按照約定的規則進行編碼，各個切片之間沒有內容的重疊或不連續，對於使用者來說，則看到無縫式平滑的串流播放。

若一份內容在編碼輸出時已提供了多種位元率，則內容切片模組會將其切割成多種位元率的串流切片。因為 Web 伺服器傳輸串流是盡可能地利用網路頻寬來進行內容的下載，沒有流量的控制機制，客戶端可以很容易地檢測到 Web 伺服器到客戶端的可用網路頻寬，從而決定下載更大或更小的媒體切片，實現位元率的自適應性。

HAS 的關鍵技術主要由兩大部分，一是內容的準備，包括了支援多台終端設備的轉碼平台以及媒體的分割切片模塊，其次是內容的傳遞，包括了基於 HTTP 的媒體來源伺服器以及面向終端設備的內容傳遞網路，完成同時發佈大量串流的服務。

2.2.3 HAS 技術特點分析

1. HAS 與其他基於 HTTP 傳輸媒體的方式一樣，和傳統的串流媒體傳遞技術相比，具有以下優勢：

    - Web 服務器更容易部署，因為 HAS 技術採用了通用的 HTTP 協定，傳統的 HTTP 緩衝/代理、防火牆等網路終端設備可以完美相容；提供了更好的相容性和到達率，可根據最後連接網路的頻寬大小動態地調整位元率，實現內容的傳遞；對於使用者來說體驗更好，且不需要媒體提供商去考慮收看使用者的頻寬。
    - HAS 除了上述優勢之外，還有以前任何技術皆不具備的特點，具體如下：
        - 使用者等待的時間更短，可以快速實現播放客戶端初始化預設選擇低位元率，開始播放後逐步向高位元率進行切換，因此，其服務質量是在可用頻寬範圍之內不斷被進行調整和優化；
        - 不需要大的緩衝記憶體，不間斷地播放，沒有抖動的平滑視頻播放體驗；

- 基於網路狀況和 CPU 解碼能力的無縫位元率切換；
- 客戶端不需要下載超過它實際消耗的內容。

綜上所述，相對於傳統的串流媒體技術，它能夠提供更好的服務質量，因為它可以使用整個可用的頻寬，而非自適應性串流技術則是強制客戶端選擇一個低於可用頻寬的固定位元率。可以預見，HAS 技術在不久的將來將得到廣泛的部署及應用。

2. HAS 的架設簡易，首先在內容準備上提供多個位元率的媒體文件，並提供一個索引文件，其中記錄了各個位元率文件的關係及特性，接下來終端設備根據初始的頻寬情況選擇一個位元率的媒體文件進行循序式播放，期間根據網路情況以及 CPU 的負載調整位元率。但是 HAS 技術方案具體實施時有許多問題需要去研究，如果這些問題沒有得到很好的解決，則無法提供最佳的使用者體驗。

2.3 影像壓縮技術

如果以每張圖像 640*480，那麼一張 24 位元的全彩 NTSC 點陣圖的大小為 640 x 480 x 3Bytes = 921600Bytes，約 0.9MB；另外，NTSC 電視系統每秒顯示 29.97 張畫面，換句話說，當電視訊號以數位化方式呈現時時，每秒大約是 29.97 x 0.9MB = 26.973MB 的資料流過，並且這還不包括聲音。如果想要在電腦上播放該視訊，電腦將需以每秒約 27MB 的速度將資料由儲存體搬出之後再搬到螢幕上；另外每小時 27MB x 3600 = 97200MB =97.2GB 的視訊容量需求是十分驚人的。如此龐大的資料儲存及傳輸量，直到近幾年硬碟技術才追趕上這樣的需求。至於現在市面上其它已上市的儲存裝置 (CD、DVD、MO、TYPE 等)，若非傳輸速度不夠，便是儲存容量不足。由此可見，數位化的資料處理雖有其好處，然而對於多媒體資料來說，未經壓縮處理的原始資料 (Raw data)，存放及傳輸仍有其問題。因此適當的對多媒體資料進行資料空間的壓縮，以方便資料儲存及傳輸實有其必要。多媒體資料壓縮技術繁多，然而在該領域中，ISO MPEG(Moving Picture Experts Groups) 提出的影音壓縮標準，可說是目前多媒體領域中的主流格式。MPEG 為一種影像壓縮和音訊壓縮格式，由 ISO 和 IEC 兩間聯盟所制定，MPEG 版本相當多樣化，

由圖 2.1可知，常見地可分為 MPEG-1、MPEG-2 和 MPEG-4，分別對應於各種不同用途 [18]。

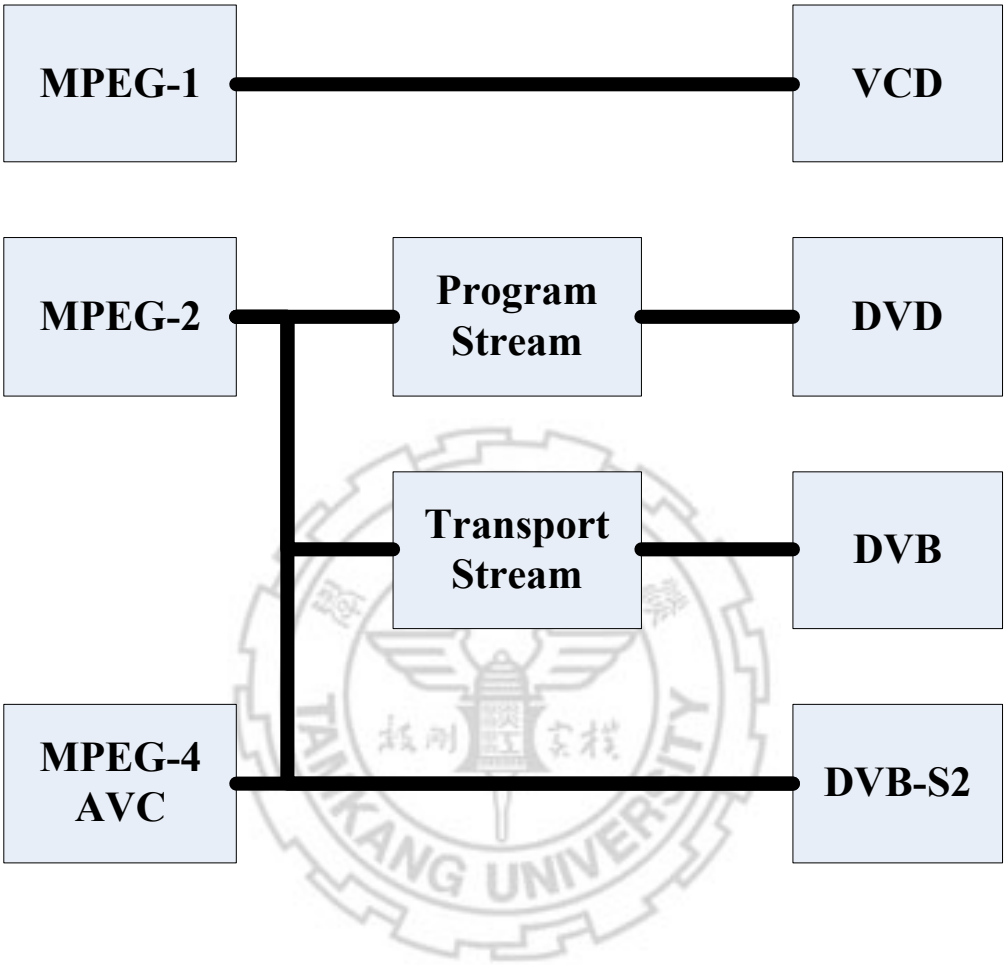

圖 2.1: 常見 MPEG 格式

2.3.1 MPEG-1

MPEG-1 是 MPEG 組織第一個制定的視訊和音訊失真壓縮標準，也是最早推出及應用在市場上的 MPEG 技術，其原來主要目標是在 CD 光碟上記錄影像，後來被廣泛應用在 VCD 光碟。視訊壓縮演算法於 1990 年定義完成。1992 年底，MPEG-1 正式被批準成為國際標準 [19]。MPEG-1 畫質雖然比起之前流通的數位壓縮視訊檔要好，然而不支援電視的交錯顯示問題，加上其畫質比起家用 VHS 系統錄影帶仍略差，更不用說是 LD。因此 MPEG-1 格式的影音 CD(Video CD，VCD) 並未受到歐美日等家電大廠重視，僅在錄放影機或 LD 不發達的地區 (如中國大

陸) 流行。

MPEG-1 是為 CD 光碟介質客製的視訊和音訊壓縮格式。一張 70 分鐘的 CD 光碟傳輸速率大約在 1.4Mbps。而 MPEG-1 採用了 Block(inter block 或 intra block) 方式達到畫面的運動補償、離散餘弦變換 (DCT)、量化等技術，並為 1.2Mbps 傳輸速率進行了最佳化。MPEG-1 隨後被 Video CD 採用作為核心技術。VCD 的解析度只有約 352×240，並使用固定的位元率 (1.15Mbps)，因此在播放快速動作的視訊時，由於資料量不足，令壓縮時 Macroblock 無法全面調整，結果使視訊畫面出現模糊的方塊。因此 MPEG-1 的輸出品質大約和傳統錄像機 VCR 相當，這也許是 Video CD 在已開發國家未獲成功的原因。MPEG-1 音訊分三代，其中最著名的第三代壓縮格式被稱為 MPEG-1 Layer 3，簡稱 MP3，目前已經成為廣泛流傳的音訊壓縮技術。MPEG-1 音訊技術在每一代之間，在保留相同的輸出品質之外，壓縮率都比上一代高。

MPEG-1 可以按照分層的概念來理解，一個 MPEG-1 視訊序列，包含多個 GOP，每個 GOP 包含多個影格，每個影格包含多個切片。影格是 MPEG-1 的一個重要基本元素，一個影格就是一個完整的顯示影像。影格的種類有四種：

1. I-圖像／影格 (節點編碼圖像，intra coded picture) 參考圖像，相當於一個固定影像，且獨立於其它的圖像類型。每個圖像群組由此類型的圖像開始。編碼時獨立編碼，僅適用影格內編碼技術，因而解碼時不參考其他影格，類似 JPEG 編碼。

2. P-圖像／影格 (預測編碼圖像，predictive coded picture) 包含來自先前的 I 或 P-畫格的差異資訊。編碼時使用運動補償和運動估計，採用前向估計，參考之前的 I-影格或者 P-影格去預測該 P 格。

3. B-圖像／影格 (前後預測編碼圖像，bidirectionally predictive coded pictures) 包含來自先前和／或之後的 I 或 P-畫格的差異資訊。編碼也使用運動補償和運動估計，預估採用前向估計、後向估計或是雙向估計，主要參考前面的或者後面的 I 格或者 P 格。

4. D-圖像／影格 (指示編碼圖像，DC direct coded picture) 用於快速進帶。僅由 DC 直流分量構造的影像，可在低位元率的時候做瀏覽用。實際編碼中很少使用。

2.3.2 MPEG-2

MPEG-2 是 MPEG 工作群組於 1994 年發行的視訊和音訊壓縮國際標準。MPEG-2 通常用來為廣播訊號提供視訊和音訊編碼，包括衛星電視、有線電視等。MPEG-2 經過少量修改後，也成為 DVD 產品的核心技術。當初 MPEG-1 設計的目地主要是影音資料於數位儲存媒體 (Digital Storage Media) 的應用，這些數位儲存媒體通常有非常低甚至趨近於零的資料傳輸錯誤；因此，MPEG-1 的系統並未設計成較強固的方式以對抗傳輸錯誤。MPEG-2 標準的目標則是希望能有更廣泛的應用，MPEG-2 的系統因此被負予了錯誤回復 (Error Resilience) 能力的使命。

MPEG-2 可分為 Program Stream(PS) 和 Transport Stream(TS) 兩種，PS 格式用於 DVD 壓縮或電腦桌機使用，Program Stream 基本上近似於 MPEG-1 的系統資料流 (System Stream)，但是內部是使用修改過的語法 (Syntax) 以及新的函式以提供一些較先近的功能。Program Stream 提供了與 MPEG-1 系統間的相容性，其解碼器 (Decoder) 設計上的要求基本上是相似於 MPEG-1 系統資料流解碼器的，一般來說，MPEG-2 的 Program Stream 解碼器是可以解 MPEG-1 的系統資料流的 [20]。而 TS 格式用於廣播電視使用，其特色為使用封包 (Packet) 儲存影像，其格式又可稱為串流，網路串流即使用 TS 格式做網路視訊的壓縮格式。Transport Stream 用於網路串流播放，其特性為會把影像封裝成一個封包 (Packet)，當在網路環境傳輸影像時，會把每個封裝成封包的影像傳輸到網路。每個封包傳輸到客戶端的時間可能都會不相同或是有可能發生封包遺失，再由客戶端重排封包順序且決定是否要重送遺失的封包。影片檔被編碼成一個個網路封包，目的端可即時接收網路封包，來達到即時的影片播放，不同於 Program Stream 需要下載完全部的檔案才可播放。每個封包都會被記錄 PID。HTTP 傳輸時，可能因為每個封包走得路由不同，而影響封包傳輸到目的端到的時間和順序都不同，此 PID 用於目的端重新排序輸出。多個封包會組合成每個 Program，其中各自記錄 MPEG-TS 所支援的各種服務，不只記錄聲音和影像，還包括了節目表，時間同步等服務。Transport Stream (傳輸串流) 簡稱 TS，主要根據 ITU-T Rec. H.262 | ISO/IEC 13818-2 and ISO/IEC 13818-3 制定而成，Transport Stream 包含了一個或以上的 Program (節目)，Program 由 Video、Audio 和節目資訊 (PSI) 組成，而 Video 和 Audio 的 Elementary Stream 將被拆解裝載到 PES Packets [21]。MPEG-2 系統資料流的抗錯誤能力主要是來自 Transport

Stream。Transport Stream 使用了新標頭及 188 bytes 固定長度的封包，固定長度的封包除了硬體較好處理之外，也更適合錯誤更正的演算。因此 Transport Stream 適合於易出錯的傳輸實體 (如有線電視網路或衛星電視) 上負載壓縮的視訊及聲音資料。事實上，Transport Stream 就是被特地設計來支援許多新功能的，如非同步多路傳輸 (Asynchronous Multiplexing)。

2.3.3 H.264

HLS 支援兩種串流影片格式，一種為 MPEG2-TS，一種為 MPEG-4 AVC(又可稱為 H.264)。

HLS 支援 H.264 用以提供更高畫質的線上串流播放。在 2001 年 12 月，ITU-T VCEG 與 ISO MPEG 共同組成聯合視訊小組 (Joint Video Term，JVT) 來研訂新的視訊壓縮格式，此新格式在 ITU-T 組織中稱為 H.264，在 ISO 組織中則納入 MPEG-4 Part-10 (ISO/IEC 14496-10) 並命名為 Advanced Video Coding (AVC)，通常合併稱為 H.264/AVC，其國際標準的第一版於 2003 年公佈，而增修的第二版也於 2005 年 3 月定案。相關研究顯示 H.264/AVC 與 MPEG-2 及 MPEG-4 相較之下，無論是壓縮率或視訊品質皆有大幅的提升，而且 H.264/AVC 也首次將視訊編碼層 (Video Coding Layer，VCL) 與網路提取層 (Network Abstraction Layer，NAL) 的概念涵蓋進來，以往視訊標準著重的是壓縮效能部分，而 H.264/AVC 包含一個內建的 NAL 網路協定適應層，藉由 NAL 來提供網路的狀態，可以讓 VCL 有更好的編解碼彈性與糾錯能力，使得 H.264/AVC 非常適用於多媒體串流 (multimedia streaming) 及行動電視 (mobile TV) 的相關應用。在第一版的標準規範中，H.264/AVC 根據使用的編碼工具種類來提供三種編碼規模 (Profile)，如表 2.1分別為 Baseline Profile、Main Profile、Extension Profile，而相對應的影片尺寸與位元率等級由 Level 1 至 Level 5.1，涵蓋小畫面與高解析度畫面的應用範圍。Baseline Profile 主要是著眼於低位元率的應用 (例如：影像通訊)，而且其運算複雜度低，所以也適合應用於個人隨身的多媒體撥放機；Main Profile 因為有支援交錯式影片 (interlaced content) 的編碼，所以適合應用於 HDTV 數位電視廣播，而且非常容易整合在傳統的 MPEG-2 Transport/Program Stream 上來傳送 H.264/AVC 位元流；對於 IP-TV 或是 MOD (Multimedia On Demand) 等應用，使用包含高抗錯性編碼工具 (error resilient

tools) 的 Extension Profile 即可以滿足這些需求。然而，微軟公司在 2003 年將其視訊壓縮技術向美國的電影電視工程師協會 (Society of Motion Picture and Television Engineers，SMPTE) 提出公開標準化的申請，並以 VC-1 (Video Codec 1) 為此新標準的命名，由於 VC-1 在高解析度影片上的表現出色，導致 H.264/AVC 在 DVD Forum 與 Blu-ray Disc Association 的高解析度 DVD 影片測試中敗陣下來，其主要原因是 H.264/AVC 使用較小尺寸的轉換公式與無法調整的量化矩陣，造成不能完整保留影像的高頻細節資訊，因此 H.264/AVC 於 2004 年展開標準增修的討論，來納入稱之為 Fidelity Range Extensions (FRExt) 的新編碼工具，並以先前 Main Profile 為基礎來擴充增加 4 個新的等級，期望能夠在高解析度影片的應用上扳回劣勢，目前增修的 H.264/AVC 第二版標準已於 2005 年 3 月發表。

表 2.1: H264 規格表

Coding Tools	Baseline Profile	Main Profile	Extended Profile
I Slice	X	X	X
P Slice	X	X	X
CAVLC	X	X	X
Slice Group And Adaptive Slice Ordering	X		X
Redundant Slice	X		X
B slice		X	
Weighted Predicition		X	X
Interlace		X	
CABAC		X	
SI and SP slice		X	X
Data partitice		X	X

H.264/AVC 標準的特色是將網路提取層的概念涵蓋進來，亦即以 NAL 封包為單位的方式來做為 VCL 編解碼的運算單位，這樣傳輸層拿到 NAL 封包之後不需要再進行切割，只需附加該傳輸協定的檔頭資訊 (adding header only) 就可以交由底層傳送出去，可以將 NAL 當成是一個專作封裝 (packaging) 的模組，用來將 VCL 壓縮過的 bitstream 封裝成適當大小的封包單位 (NAL-unit)，並在 NAL-unit Header 中的 NAL-unit Type 欄位記載此封包的型式，每種型式分別對應到 VCL 中不同的編解碼工具。NAL 另外一個重要的功能為當網路發生壅塞而導致封包錯誤或接收次序錯亂 (out-of-order) 的狀況時，傳輸層協定會在 Reference Flag 作設定的動作，接收端的 VCL 在收到這種 NAL 封包時，就知道要進行所謂的糾錯運算 (error concealment)，在解壓縮的同時也會嘗試將錯誤修正回來。一個完整

的 H.264/AVC bitstream 是由多個 NAL-units 所組成的，所以此 bitstream 也稱之為 NAL unit stream。一個 NAL unit stream 內可以包含多個壓縮視訊序列 (coded video sequence)，一個單獨的壓縮視訊序列代表一部視訊影片，而壓縮視訊序列又是由多個 access units 所組成。當接收端收到一個 access unit 後，可以完整地解碼成單張的畫面，而每個壓縮視訊序列的第一個 access unit 必須為 Instantaneous Decoding Refresh (IDR) access unit。IDR access unit 的內容全是採用 intra-prediction 編碼，所以自己本身即可完全解碼，不用參考其他 access unit 的資料。access unit 亦是由多個 NAL-units 所組成，標準中總共規範 12 種的 NAL-unit 型式，這些可以進一步分類成 VCL NAL-unit 及 non-VCL NAL-unit。所謂的 VCL NAL-unit 純粹是壓縮影像的內容；而所謂的 non-VCL NAL-unit 則有兩種：Parameter Sets 與 Supplemental Enhancement Information (SEI)，SEI 可以存放影片簡介、版權宣告、使用者自行定義的資料…等；Parameter Sets 主要是描述整個壓縮視訊序列的參數，例如：長寬比例、影像顯現的時間點 (timestamp)、相關解碼所需的參數…等。這些資訊非常重要，萬一在傳送的過程中發生錯誤，會導致整段影片無法解碼。以往像 MPEG-2/-4 都把這些資訊放在一般的 packet header，所以很容易隨著 packet loss 而消失。現在 H.264/AVC 將這些資訊獨立出來成為特殊的 parameter set，可以採用所謂的 out-of-band 的方式來傳送，以便將 out-of-band channel 用最高層級的通道編碼 (channel coding) 保護機制，來保證傳輸的正確性。

由於 H.264/AVC 在視訊編碼演算法上的改進，其壓縮比及視訊品質與 MPEG-2/-4 相較下有大幅度的提昇，而其 NAL 概念有助於在有限頻寬的傳輸通道上來傳送高品質的視訊內容。對於高畫質數位電視或高畫質 DVD，以 H.264/AVC 的編碼技術都可以很輕易地滿足應用需求，但就市場面來看，VC-1 標準憑藉著微軟在 PC 平台的優勢與低價授權的策略，所以當時 VC-1 是 H.264/AVC 的強大挑戰者。

2.4 Adobe Flash Dynamic Streaming

由於目前主流趨勢是使用 HTTP 協定來傳輸串流檔案，因此 Adobe 開發了 HTTP Dynamic Streaming 協定來取代過去使用 RTMP 協定來傳輸串流的方法，HDS 協定的架構大致上與 Apple 的 HTTP Live Streaming 協定類似。

和 HTTP Live Streaming 協定類似，在 HDS 中分為播放清單檔以及多媒體檔。

HDS 的播放清單檔稱為 manifest(副檔名.f4m)，是採用 XML 格式撰寫的描述檔，除了可用的串流外還有相關資訊如 DRM 等。在 HDS 中也使用兩層式的架構，第一層清單 (set-level manifest) 中會描述可用的不同品質 (位元率) 的串流以及對應的第二層清單，其中列出了各個段落的影像以及位址，再由播放器以 HTTP 要求取得檔案後播放。而多媒體檔案方面，在 HDS 中可使用的影音編碼為 H.264/AAC 或是 VP6/MP3，封裝格式是 MP4 格式的延伸格式 (副檔名.f4f)。在伺服器端完整的串流是以一個檔案儲存，當要傳送至客戶端時才會依據客戶端要求的時間段落 (time code) 將該段落切割為獨立檔案傳送。

要使用 HDS 串流，伺服器端需要使用 Adobe 公司的 Flash Media Server 4.0 以上版本，以及其他相對應的如 Flash Media Live Encoder 軟體。而在發佈端由於是將準備好的清單以及串流檔案放置到伺服器上讓客戶端下載，因此和 HTTP Live Streaming 類似，使用 Apache 等 HTTP 伺服器即可，同時也可以利用 CDN 等 HTTP 的快取機制。要觀賞 HDS 串流，客戶端需要安裝 Adobe Flash Player 10.1 以上版本的外掛程式，但是在新版本的 Flash Media Server 4.5 中，由於伺服器可以將串流轉換為架構類似的 HTTP Live Streaming，因此可以在支援 HTTP Live Streaming 播放的客戶端中播放。

2.5 Microsoft Smooth Streaming

Smooth Streaming，正式名稱為 IIS Smooth Streaming，是由微軟公司為 Internet Information Services 7.0 4 發佈的擴充功能。Smooth Streaming 技術雛形最早的應用是在 NBC 為 2008 年北京奧運架設的網站上，而正式發表的 Smooth Streaming 技術則是以當時使用的技術加入更完整的管理等功能。

Smooth Streaming 使用 MPEG-4 Part 14(ISO/IEC 14496-12) 作為檔案格式 (副檔名.mp4)，支援的影像壓縮編碼為 VC-17 以及 H.264。與 HTTP Live Streaming 不同的是，在 Smooth Streaming 伺服器中各個編碼品質都是完整的一個檔案。當客戶端要求串流時，伺服器會根據要求的串流時間切割虛擬片段 (virtual chunks) 並儲存為獨立檔案傳送給客戶端。

要播放 Smooth Streaming，可以使用微軟開發的 Silverlight 開發框架撰寫播放器嵌於網頁中以瀏覽器開啟、XBOX、Windows Phone 7，或者是透過 IIS Media

Services 4 的新功能，將使用 H.264 編碼的 Smooth Streaming 轉換為 HTTP Live Streaming 相容的格式 (如 M3U8 清單、MPEG-2 TS 封裝)，以 HTTP Live Streaming 播放器播放。

由於 Smooth Streaming 的伺服器端需要安裝 IIS 7.0 以上的版本，因此僅限於 Microsoft Windows 平台可以使用，而且在發佈前需要經過許多的設定步驟。由於這些限制，Smooth Streaming 的伺服器端無法像 HTTP Live Streaming 一樣使用各種一般 HTTP 伺服器建置，也無法整合雲端儲存型的 CDN 服務以減少延遲。除此之外，即使 Smooth Streaming 看起來可以在較多瀏覽器及平台觀賞，但由於在瀏覽器需要使用 Silverlight 外掛，因此與本論文希望使用未來有機會成為標準化、跨平台不需安裝外掛的技術目的衝突。

2.6 MPEG Dynamic Adaptive Streaming over HTTP

Dynamic Adaptive Streaming over HTTP(DASH) 是由 MPEG 組織所發展的 adaptive bitrate streaming 標準，DASH 同樣支援隨選視訊以及即時串流、動態調整串流品質，並參考了許多已有的協定的優點。其標準制定是透過公開徵求各界提供提案，再由其中選擇、統整出適當的草案。相較於前面的幾個協定，2010 年才開始發展的 DASH 雖然發展時間較短，但已經在 2011 年 12 月成為網際網路標準，目前 DASH 標準已經受到業界各公司如 Apple、Microsoft、Adobe 及其他重要廠商承諾未來會為其提供支援，但目前還沒有任何瀏覽器支援播放 DASH 串流，可使用的客戶端軟體只有 VLC 播放程式。

圖 2.2表示 MPEG DASH 的客戶端在播放串流時和 HTTP 伺服器之間的溝通，在 DASH 中的檔案有兩個部分:Media Presentation Description (MPD) 以及 segment 也就是實際的影音檔案。MPD 檔案，類似於 HLS 協定中的 index files，但其中描述的文字格式為 XML，描述了可用的內容、針對於該內容的各種不同品質串流以及其網址等資訊。而針對各種不同品質的實際 segment 內容檔則是同時可以支援以單一檔案透過時間分割段落，或者是實際分割為多個檔案的方式。

要播放 DASH 串流時，客戶端會先分析取得的 MPD 檔案，進而得到相關的時間、可取得串流、編碼、解析度或是 DRM 等資訊，接著再以 HTTP 取得適當的串流內容。在播放時，DASH 客戶端同樣會持續的監控網路品質，並調整取得的

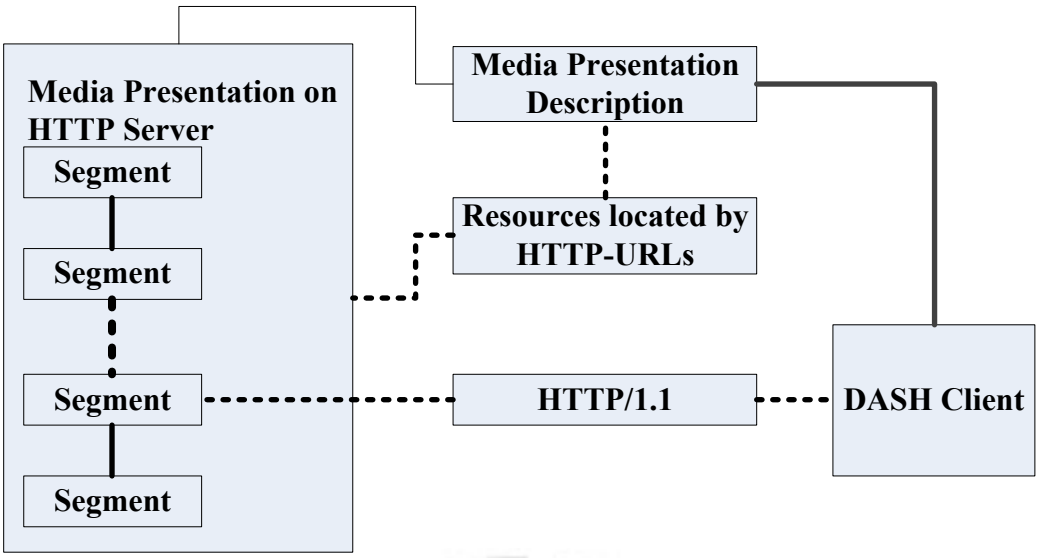

圖 2.2: DASH 運作示意圖

串流品質。與 HTTP Live Streaming 不同，DASH 中影像以及音效是分離儲存、傳送再由客戶端組合播放的，因此可以支援不同語言配音的選擇、切換。除此之外，MPD 中還記載了許多資訊，如同一時間不同角度的影像、或是各種字幕，與其他的協定不同，DASH 中只定義了 MPD 和 segment 的格式，但對於 segment 中的影音檔要使用什麼編碼格式卻沒有定義。因此影像可以使用 H.264 或是 WebM，音訊可以使用 MP3、AAC 等編碼格式，甚至在包裝格式方面也可以選擇 MPEG-2 TS 或是 MP4 格式。這種開放性對於期待 DASH 標準可以解決 HTML5 標準中未定義影音編碼的問題來說是讓人失望的，就像目前 HTML5 一樣，要完整支援 DASH 標準的客戶端需要同時支援許多種的編碼格式。同樣的 DASH 中對於一個 DASH 客戶端該有的功能和實作的方式也沒有定義。

2.7 Apple HTTP Live Streaming

早期提供網路串流，需要在電腦安裝同樣的伺服端和客戶端的程式，由伺服端依照特定的影像編碼後，透過 TCP 或 UDP 或 RTP/RTCP 等相同的傳輸協定，傳送影像和聲音給客戶端，客戶端再利用相同的編碼格式來進行解碼。現在是雲端的時代，應該提供使用者更簡便的方式收看線上串流，不應該由使用者去煩惱該

安裝什麼播放程式，該選擇那種播放格式。如果客戶端只需要打開瀏覽器即可使用各種服務，可提供使用者更方便的服務，簡化使用者操作上的問題，提升使用者的觀看品質。

傳統使用 RTP 協定的線上串流技術，因為使用特殊協定和通訊埠，除非防火牆特別為了此協定開啟通訊埠，不然容易被防火牆阻擋。且 RTP 協定地設計是當播放線上串流時，其會一直保持連線，所以伺服端要管理每一個連線的客戶端，且各自進行單一獨立的 Streaming Session，其會耗費相當大的伺服器資源。所以現在利用 HTTP 協定傳輸每一個封包給客戶端，伺服端只要循序處理每個封包，對伺服端的負載相對減少，且可解決以上防火牆阻擋的問題，因為 HTTP 協定所使用的通訊埠，是網頁伺服器一定要開啟的通訊埠。

Adobe Flash Dynamic Streaming 和 Microsoft Smooth Streaming，雖然提供使用者只需經由瀏覽器即可觀看網路視訊串流，但由於瀏覽器本身不支援此技術，所以需要額外安裝外掛套件，對使用者來說相對不方便。而現今移動式裝置、智慧型手機和平板的市場越來越大，但受限於移動式裝置普遍無法在瀏覽器安裝外掛套件，所以觀看網路視訊串流，多利用傳統模式經由安裝應用程式和伺服器保持連線，用以傳輸視訊至客戶端。而 Apple HTTP Live Streaming 是少數可由瀏覽器支援，用以觀看網路串流。

HLS 和其他 RTP/RTSP 等線上串流技術不同處，在於 HLS 使用 HTTP 協定，不使用自訂的網路協定，所以不容易被系統防火牆或代理伺服器所阻擋。且由於使用 HTTP 協定，所以使用 TCP 傳輸資料，可確保影音傳輸品質和正確性。而且 HLS 讓架設者可快速架設出隨選視訊的線上串流系統，架設者可以使用任何一種常見的網頁伺服器架設，架設方法簡易，所以本篇論文才選擇 HLS 為實現理論的技術。

HTTP Live Streaming 傳輸是走 HTTP 的協定，走 HTTP 的第一個好處為可使用瀏覽器傳輸視訊，開發者可以在不讓客戶端安裝軟體下，而直接作服務網頁的修改，可方便開發者開發軟體；第二個好處是比較不容易被路由器的防火牆所阻擋，當客戶端處於路由器後端，還是可以經由瀏覽器觀看網路串流，而不需要特別去設定路由器的防火牆規則設定。

HTTP Live Streaming 架構圖如圖 2.3，當視訊錄製進來時，需要壓縮編碼成

MPEG2-TS，然後會把 Transport Stream 切割成多個時間短且檔案小的 Transport Stream。Apple 公司建議值為 10 秒一個 Transport Stream，然後會產生 M3U8 格式的播放清單列表，儲存所有 Transport Stream 的 URL，對客戶端只需在瀏覽器開啟其 M3U8 播放清單即可播放，也可把 M3U8 播放清單嵌入至 HTML5 的網頁。HTTP Live Streaming 可經由 M3U8 播放清單，提供多種畫質格式，其利用 M3U8 索引檔為區分各種不同畫質，客戶端可依據頻寬變化，選擇適合的播放畫質來播放。

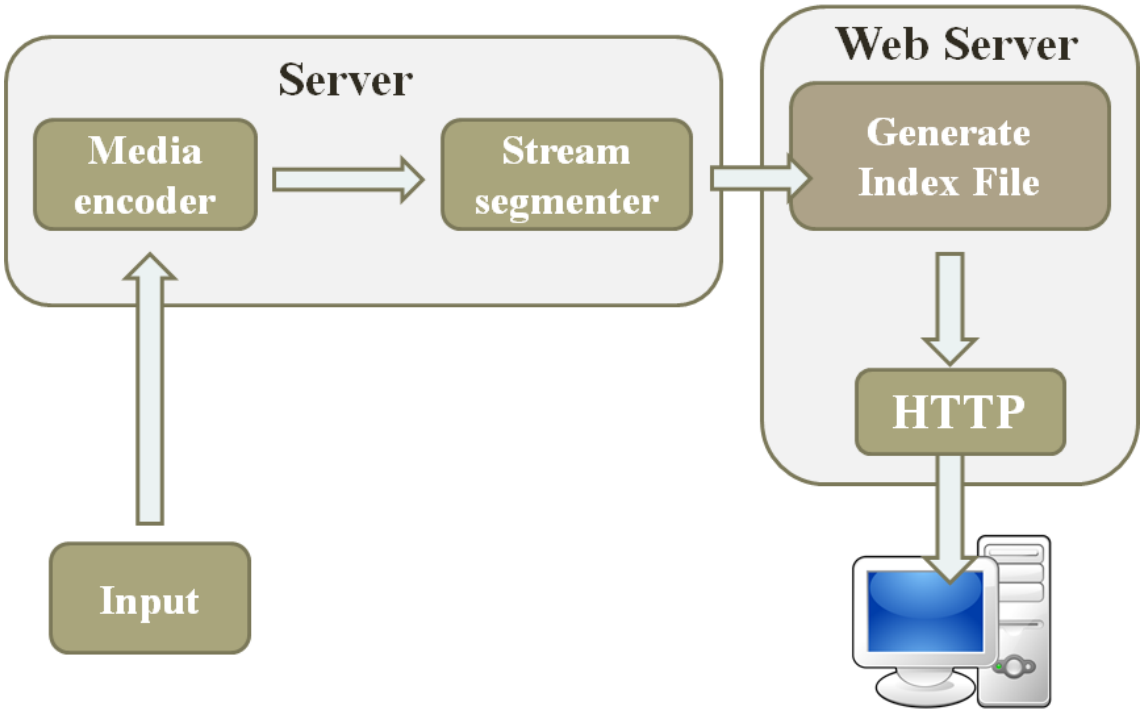

圖 2.3: HLS 架構圖

其工作原理為把一完整的串流檔 (Transport Stream)，切割成一個個時間短的串流檔，每次只需下載一個串流檔，即可開始進行線上串流且提供給使用者。由於每個串流檔，時間很短且檔案長度短，使用者可以很快觀看到影片，可減少線上串流播放的延遲時間長度。且當使用者經由瀏覽器播放線上串流時，伺服端可提供多種畫質或各種不同位元傳輸率 (Bit Rate) 的串流檔，由瀏覽器選擇適合頻寬的檔案，瀏覽器會依據頻寬，動態調整畫質，可讓使用者不容易因為頻寬的變化，影響到播放的流暢性。使用者能利用瀏覽器經由 HLS 播放影片或音樂，是利用 HTML5 在其中加入讀取 M3U8 檔，使用者即可經由 M3U8 檔播放選取的線上串

流。

2.7.1 M3U8 播放清單

用於讓瀏覽器可以循序式播放 Transport Stream，其格式如圖 2.4，標籤 EXT-X-TARGETDURATION 設定此播放清單每個 Transport Stream 播放時間，標籤 EXTINF 可設定個別 Transport Stream 的時間長短，每個標籤 EXTINF 下，都會連接 Transport Stream 的 URL。

```
 #EXTM3U
#EXT-X-TARGETDURATION:10
#EXT-X-MEDIA-SEQUENCE:1
#EXTINF:10,
http://url/segment0.ts
#EXTINF:10,
http://url/segment1.ts
#EXTINF:10,
http://url/segment2.ts
#EXT-X-ENDLIST
```

圖 2.4: M3U8 格式

圖 2.5為索引檔，HTTP Live Streaming 可利用索引檔，提供多種播放畫質，標籤 EXT-X-STREAM-INF 可設定頻寬為多少時，即播放相對應的 M3U8 播放清單，由客戶端去決定該如何選擇播放影片。

```
   #EXTM3U
#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=200000
http://ALPHA.mycompany.com/lo/prog_index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=200000
http://BETA.mycompany.com/lo/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=500000
http://ALPHA.mycompany.com/md/prog_index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=500000
http://BETA.mycompany.com/md/prog_index.m3u8
```

圖 2.5: M3U8 索引檔

HTTP Live Streaming 常用於隨選視訊，其影片是已經錄製編碼肛儲存在伺服器端，特點在於可隨時調整時間軸至想播放的時間，不用等全部影片讀取完才

可播放，可讓客戶端使用者等待時間減少，而提升觀看體驗。但如果輸入視訊為即時影像，如網路攝影機，其只能即時產生一種輸出格式，客戶端反而無法利用 M3U8 索引檔依據頻寬變化，決定該如何使用播放畫質。

2.7.2 HTML5 和 FLASH

現在行動裝置當道，使用者逐漸轉移使用習慣，很多時候使用者會選擇利用手上的行動裝置解決各種需求，所以各大線上串流網站開始提供使用者在行動裝置上播放線上串流影片。傳統網站需要播放瀏覽器不支援的格式，都需要安裝各種外掛程式，而現今最常被應用於線上播放影片和音樂得外掛程式，即為 Adobe 公司的 Flash。但因為大部份的行動裝置得瀏覽器皆不原生支援 Flash，且 Flash 相對較耗電，不適用於行動裝置上。現今只剩 Android 平台可以額外自行安裝 Flash，不過效能一直不佳也是一大問題，所以很多線上串流網站無法直接利用網頁提供服務，所以都選擇在行動裝置上撰寫客戶端軟體，提供使用者連到線上串流伺服器下載且播放影片或音樂。

為了解決瀏覽器一定需要使用 Adobe Flash 等外掛程式方可播放影片或音樂，HTML5 提供新的解決方案，直接由瀏覽器內建播放器。網頁開發者只需要撰寫簡易的 HTML5 程式碼，讓使用者不需額外安裝軟體，即可順利播放瀏覽器可支援得影片或音樂。

HTML 為瀏覽器產生網頁的一種排版語言。HTML 1.0 於 1993 年發展至今，已進步到 HTML 5.0，相較 HTML 4.0 以前在網路影音播放發展沒像現在如此發達，所以在影音播放的支援不足，所以需要幫瀏覽器安裝各種播放外掛，例如 Adobe 的 Flash，Microsoft 的 Sliverlight 或是 Sun 的 Java 等，才可在網頁上播放所使用影音格式。而在 HTML 5.0 改善了對影音播放的支援特性，由瀏覽器原生支援特定影音格式，且 HTML5 支援 MPEG2 和 MPEG4 等眾多影音播放格式，現今使用的瀏覽器都已支援 HTML5，例如 Google Chrome，Apple Safari，Mozilla Firefox 和 Microsoft Internet Explorer 等 [22] [23]。

2.7.3 字幕檔 (WebVTT)

現在瀏覽器經由 HTML5 的支援，已經可以不需要額外安裝外掛程式來播放影片。而觀看國外影片時，不是每個人都能聽懂影片內容，所以才有了字幕的存在。關於 HTML5 視頻字幕最初是 HTML5 標準的一部分，定義為一種 WebSRT 文件格式，這種格式可以通過使用通常的 SRT 文件格式被用來指定視頻字幕，而後命名為 WebVTT，字幕標準源於 HTML5 標準，然後自己成為一個標準體系 [24] [25]。

如圖 2.6，WebVTT 是一個簡單的文字檔，其副檔名為.vtt，使用 UTF-8 編碼，所以可以支援多國語系字幕顯示，主要記錄著兩種資訊，一為時間軸，一為字幕內容。檔案第一行開頭是 WEBVTT 標記，後面連接換行符號。計時提示的格式為 HH:MM:SS.sss。開始和結束提示是由一個空格、兩個連字號和一個大於符號 (-->) 再加上另一個空格來分隔。計時提示自成一行並搭配一個換行符號。緊接著提示的是輔助字幕文字。文字輔助字幕可以成一或多行。唯一的限制是文字行之間不可以有空白行，WebVTT 檔案的 MIME 類型是"text/vtt"。

```
WEBVTT

00:01:12.322 --> 00:01:14.407
<i>在這個特殊的日子</i>

00:01:14.491 --> 00:01:16.743
<i>波士頓的孩子們聚集起來</i>

00:01:16.910 --> 00:01:18.662
<i>一起欺負猶太孩子</i>
```

圖 2.6: WebVTT 字幕檔

```
<video width="640" height="480" controls>
<source src="video.mp4" type="video/mp4" />
<source src="vidoem.webm" type="vidoe/webm" />
<track src="subtitles.vtt" srclang="zh" label="Chinese" />
</video>
```

圖 2.7: WebVTT 在 HTML5 格式

如圖 2.7，WebVTT 要結合 HTML5 輸出字幕至視訊串流中，需要使用 track tag 插入 vtt 字幕檔，由於字幕檔中時間軸的設定，會相對應配合網路視訊串流時間

軸，達到觀看視訊時，可以顯示字幕在影片上 [24]。在 HTTP Live Streaming 由於影片被壓縮切割成無數個時間短的影片，原始 WebVTT 無法完全適用於 HLS，因為無法利用單一字幕檔中的時間軸同步所有影片檔，所以需要為每個影片檔各別提供 WebVTT 字幕檔，WebVTT 字幕檔即可順利播放顯示於 HLS 的影片串流中 [26]。

第三章　Adaptive Transcoding HTTP Live Streaming System

隨選視訊用途在於提供使用者可以觀看預錄影片，所以已經是事先處理好的影片，好處在於可隨時跳轉影片，觀看使用者自行想看的影片位置。不過壞處在於需要大量的儲存空間，存放所有伺服器提供給使用者的影片檔，而且當頻寬不足播放此影片時，容易產生播放停頓，影響使用者體驗。

即時線上串流在於提供給使用者觀看線上轉播等即時輸入訊號，好處是使用者可以即時觀看影像，例如使用者可在網路上即時觀看籃球比賽，而使用者觀看比賽最忌諱看到一半突然停格；或是觀看到投籃畫面其實是一段時間之前的影像，和電視觀看到的影像慢很多，會減少使用者願意使用此服務的體驗。所以即時線上串流最重要為減少影像延遲問題，且不用等球賽結束後，才可以觀看重播球賽影片。就以伺服器角度來講，也可大量減少儲存空間。但伺服器需要提供大量的運算能力，用以應付大量的即時轉碼問題，且當多使用者連入，將大量考驗伺服器運算能力，且最重要為影像輸出位元率為固定值，如果頻寬不足時，完全無法播放即時串流。

以上兩種都產生相似的問題，容易受到頻寬的變化且容易受到多使用者連入，影響播放的流暢度。本篇論文在於解決以上傳統遇到的問題，提出一種合乎現在網路環境狀況的模型。

標準 HLS 適用於隨選視訊上，瀏覽器如需要支援動態調節影片位元率，必須在伺服器為同一個影片準備多個不同位元率的影片檔，且需要在 M3U8 檔案中，設定各種條件，當某個頻寬區間，瀏覽器該選擇下載且播放某個位元率的影片，用以達成調節影片畫質 [27]。如未在 M3U8 檔案中設定頻寬條件，會經由瀏覽器頻寬調整演算法去選擇適合畫質影片。由於其頻寬選擇演算法為 Greedy，所以會產生一種問題，當有新的連線進來時，由 Greedy 演算法，每個連線都會試圖取得最大畫質設定，但如果伺服器頻寬無法讓全部連線保持現有畫質設定時，每個連線的瀏覽器最後會選擇把畫質都設定為當時播放串流提供的最低畫質，在下一個播放區間時 (10 秒)，會把所有連線都往上調高一個畫質，如果還是超過現有總頻

寬，又會全部連線設定為最低畫質。一直反覆測試與設定，其產生問題為影響到全部使用者的觀看畫質。論文 [28] 提出一個解決方案，在媒體伺服器和使用者中間加上一個頻寬調節伺服器，當有新連線進入時，先由頻寬調節伺服器經由其頻寬選擇演算法，找出每個連線在現有頻寬最佳畫質，然後才開始提供播放服務，所以當新連線開始播放影片時，所有連線都被設定為最適合畫質，可解決以上問題。

但論文 [28] 會延伸出幾個問題，第一為此解決方案只適用於隨選視訊服務，只有隨選視訊服務才可提供所有影片的各種畫質設定值，用以提供頻寬調節伺服器計算出每個連線最適畫質。第二為論文中未詳述頻寬偵測方式，而頻寬偵測需要考慮兩種頻寬，其一為伺服器可提供總頻寬，其二為伺服器和客戶端最大頻寬，或指客戶端網路連線最大頻寬。如果頻寬調節伺服器計算出的最適頻寬，超過客戶端網路最大連線頻寬，一樣會產生客戶端播放串流的延遲問題。

本論文將提出 Adaptive Transcoding HTTP Live Streaming System(AHLS)，用以解決以上問題，以下章節將分為說明以上問題的理論基礎和 AHLS 實作。

3.1 理論基礎

3.1.1 頻寬偵測

頻寬偵測在本論文為一重要課題，需要測定出伺服端和客戶端之間頻寬變化，論文 [29] 提出改善 Pathchar 演算法來達成效能更好的頻寬偵測。Pathchar 演算法由美國學者 Van Jacobson 於 1997 年 MSRI 中提出 [30]，且其提出 traceroute 工具用以實作 Pathchar 演算法，現已成為最常見用於頻寬偵測工具。其演算法提出網路中發送不同大小的測量封包，根據每個節點回應時間 (Round-Trip Time) 測量出每個節點之間頻寬。但延伸出一個問題，由於 Pathchar 可偵測出每個節點頻寬，如果節點越多，其測量頻寬時間過久。雖然論文 [29] 改善 Pathchar 演算法，但其功能和效能皆不適用於本論文，本論文講求即時頻寬偵測且也無需要偵測出每個節點頻寬，只需要能即時偵測出伺服端和客戶端之間頻寬。

HTTP adaptive streaming 有兩個主要重點，提供畫質的最佳化和縮短從影像資源傳入至伺服器，而後傳輸至客戶端且播放，其中間的時間差，此為一種

end-to-end(e2e) delay [31]。標準 HLS 設計，由於視訊需要壓縮切割成時間固定的串流檔，再傳輸至客戶端，不同於 RTSP 架構，無法即時邊壓縮邊傳輸串流，所以一定會產生固定的 e2e delay。如何偵測頻寬，但不提高 e2e delay 數值，其為重點課題。本論文提出在之後實作中，於伺服端和客戶端中，加入一個虛擬中間層，用以替代伺服端與客戶端連接，當客戶端要求下載 HLS 串流檔時，改由虛擬中間層與客戶端連接且傳輸串流，即可測量出每個串流切片伺服端與客戶端傳輸時間。而每個串流檔檔案大小為已知條件，ε_s 為檔案大小，τ_0 為串流檔傳輸時間，由公式 3.1即可計算出伺服端與客戶端現有頻寬 β。

$$\beta = \frac{\varepsilon_s}{\tau_0}, \tag{3.1}$$

3.1.2 動態調節演算法

標準 HLS 架構為把串流切割成無數個時間短的影片串流檔，當網路頻寬變化大時，將在下個時間切片切換畫質，用來調整串流位元率大小。本論文提出經由頻寬偵測得出現有頻寬後，在伺服端即時壓縮且切割輸出的串流檔，由於已知每個連線現在頻寬，即可把此數值，當作串流壓縮時的最大參數，確保下個時間點的串流檔位元率不會超過現在頻寬量。

何謂位元率 (BitRate)，位元率是用於判定串流在一秒中傳輸量，用於表示串流的品質，當一秒內傳輸資料越多，表示位元率越大，即表示其串流品質越好，反之位元率越小，表示串流品質越差。在視訊壓縮時，位元率無法當作一個壓縮的設定參數，需要調整視訊各種參數，用以合乎設定位元率。

以 MPEG 壓縮為例，MPEG 壓縮把畫面分為 I、B、P 三種畫格 (Frame) 表示方式，經由三種畫格儲存序列統稱為 GOP(Group of pictures)，其排序方式為 I、B、P。I 畫格表示儲存完整的視訊畫面，I 畫格的編碼方式是採用類似於 JPEG DCT 的方式處理，它並不考慮與其他畫面之間的關係，只有使用本身的資料進行編碼，所儲存的是一張完整的畫面。經量化後 (Quantization)，再經可變動編碼 (Variable Length Coding)，由於完整記錄畫格的資料，因此壓縮率比較差，其壓縮率大約為 1:7。I 畫格是一個序列或影片群組的第一張，接著是影片群組的 B 與 P 畫格，它們都會參考到 I 畫格的資料，所以在網路傳輸時，需要保護該畫格的損壞遺失，

以免讓其它畫格參考了不正確的資料，發生馬賽克的現象。P 畫格在編碼時，會參考 I 畫格，只記錄與前一張 I 畫格的差異，影像壓縮率次高為 1:20。而 B 畫格於 GOP 儲存位置介於 I 畫格與 P 畫格之間，其畫格在編碼時，使用雙向的參考前後的 I 畫格與 P 畫格，而參考的位置以移動預測，所產生的移動向量來表示。由於參考前一個 I 畫格或 P 畫格，且以動態預測補償編碼，若找不到最適合的大區塊時，則使用 Intra 模式編碼 (I-Frame)，本身不做為其它畫面的參考用，擁有最高的編碼壓縮率為 1:50，所以影像儲存容量以 I 畫格最大，B 畫格最小。

M 與 N 是用來描述一 GOP 結構的參數，M Frame = 前一段連續的結束 P 與後一段連續的結束 P 畫格的距離數量，例如：M=3，其表示為 IBBP，M=4，其表示為 IBBBP，M=5，其表示為 IBBBBP，由此類推。N Frame = 在一組 GOP 裡的畫格總數量，例如：M=3，N=15，其 GOP 排序為 IBBPBBPBBPBBPBBPIBBPBBPBBPBBPBBP。所以當增加 B 畫格與 P 畫格數量，即可有效減少 MPEG 壓縮的位元率，雖然增加 B 或 P 畫格可有效降低串流位元率，如果串流為高速移動影片，在影像解碼還原時，容易產生影像模糊不清、畫面抖動或影像殘影，因為 B 與 P 畫格不是完整記錄畫面，只記錄與其他畫格差異。而在 H.264 編碼中，其修改了 DCT 的轉換公式，從浮點數 DCT 轉換，改由整數 DCT 轉換，在逆轉換時不會產生因為浮點數運算導致的有效位數誤差問題，且熵編碼可以使用可變長度編碼。H.264 全部資料都從浮點數儲存改成整數儲存，可以有效減少位元率大小。

以 FFmpeg 為例，其調整位元率方式為測量 1 秒的串流中，在調整 GOP 和解析度後，是否有達到由位元率計算出設定的檔案大小，如果沒有達成，會重新修改 GOP 參數和解析度，以求達成設定的位元率，所以表示位元率越低，其畫面品質和解析度都會越差。

3.1.3 編碼壓縮模型

經由記錄串流檔下載時間，即可測量出現在頻寬。當客戶端開始播放串流檔時，利用此空閒時間，伺服端利用測量出的頻寬，當成下個串流檔壓縮時的轉換參數。

轉碼器 (Transcoder)，其可分為兩種功能，一為編碼 (Encode)，一為解碼 (Decode)。如果影片來源不是 HLS 原生支援的 MPEG2-TS 或 MPEG4-TS (H.264)，

一定要先經由解碼器 (Decoder) 還原成未壓縮格式，然後編碼器 (Encode) 才可轉換為串流檔。本章節將提出兩種轉碼器的模型，HLS 設計是每十秒編碼成一個 TS 檔。客戶端播放影片時，傳輸下一個 TS 檔給客戶端。

影片傳輸給客戶端時，經由計算下載時間，用以測量出頻寬變化。由圖 3.1和表 3.1得知，每次下個串流檔開始壓縮的時間點，一定為每次傳輸完串流檔才能開始壓縮，但瀏覽器在播放到影片的一半時間點 (5 秒)，將會請求下一個串流檔，每次轉碼下一個串流檔時間為影片時間的一半 (5 秒)，所以轉碼與傳輸串流檔至客戶端，總共只有 5 秒的處理時間，將會產生一個問題，無法處理更高品質的影片，因為當伺服器無法在有限時間內壓縮完下一個串流檔，將會影響後面串流的下載與播放，客戶端將會再等待下個時間區間 (10 秒)，才會再要求下載下一次串流檔，而此狀況會導致觀看延遲而影響使用者觀看體驗。

表 3.1: 第一版編碼模型參數

參數	說明
R_0	要求第一個串流檔
B_0	回傳第一個串流檔且測量頻寬
T_1	要求編碼且代入測量頻寬為參數
P_0	播放串流
R_1	要求下一個串流檔
B_1	回傳下一個串流檔且測量頻寬

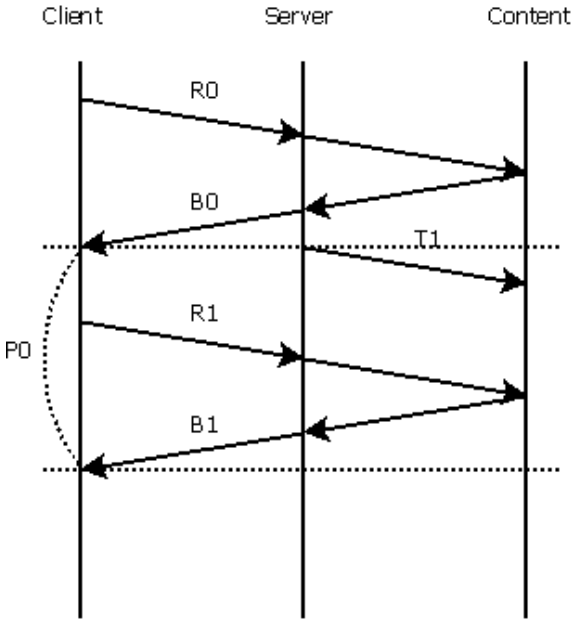

圖 3.1: 第一版編碼模型

為了解決以上問題，用以加速轉碼的速度，將提出第二種轉碼器模型，如圖 3.2和表 3.2，把轉碼分成編碼 (Encode) 和解碼 (Decode) 來處理。當上一個串流檔還未下載完時，伺服端即可開始進行下個串流檔解碼 (Decode)，由於解碼不需要等待頻寬偵測結束，當客戶端下載完串流檔且測量得知現在頻寬，客戶端開始播放影片時，伺服端進行下個串流檔編碼 (Encode)。此模型與第一種模型先解碼再編碼的流程不同，此方式可以令轉碼器可處理時間拉長至十秒。因為在測量完頻寬至客戶端要求下一個串流檔之間的時間區間，只需要處理編碼 (Encode) 成串流檔。

表 3.2: 第二版編碼模型參數

參數	說明
R_0	要求第一個串流檔
B_0	回傳第一個串流檔且測量頻寬
D_1	先行解碼第一個串流檔
E_1	壓縮第一個串流檔
P_0	播放串流
R_1	要求下一個串流檔
B_1	回傳下一個串流檔且測量頻寬
D_2	先行解碼第二個串流檔
E_2	壓縮第二個串流檔

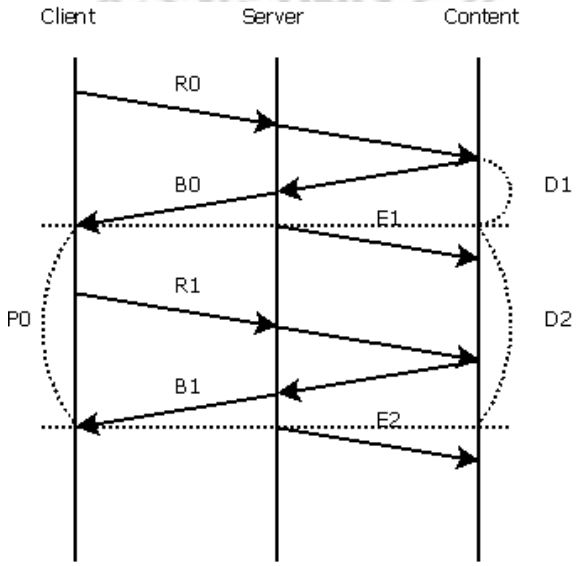

圖 3.2: 第二版編碼模型

3.2 實作

現今行動裝置興起，使用者常使用行動網路或無線網路上網，除了網路瀏覽外，利用行動網路觀看線上串流，也逐漸普及。但行動網路容易受很多因素影響傳輸速度，導致在線上串流傳輸不穩定，不定時的播放停格，影響使用者觀看感受，不合乎在外快速觀看影片的習慣。

在實作方面，在行動裝置觀看線上串流，大多需要安裝播放程式，用於連線到伺服器播放影片。如果使用者只需利用瀏覽器播放影片，是否更為方便？讓線上串流提供商，不用考慮各平台相容性問題和是否需要花時間撰寫 App，專心撰寫網頁和提供更多的線上串流服務。

本章節將會分別針對以下幾種方向，將會分別介紹實作中，所有會使用到的軟體，該如何架設應用於此論文實作且會遇到些什麼問題，該如何解決此問題，提出所有設計時遇到問題且該如何改善，用以合乎使用者需求的設計。

3.2.1 系統環境架設

以下將詳述系統環境的架設：

1. 安裝網頁伺服器且需支援 PHP，以本篇實作為例，其安裝 Apache Web Server，且安裝了 PHP 的模組。
2. 安裝資料庫系統且建立相關表格，以本篇實作為例，其安裝 MySQL 在系統中
3. 複製實作的程式碼至網頁伺服器預設目錄

當以上工作都架設完成，使用者只需將想播放影片放置於影片資料夾中，即會自動更新檔案清單資料庫，馬上可以提供給使用者播放想觀看的影片。本篇論文實作出跨平台的線上串流播放系統，可同時使用行動裝置和在電腦上播放，例如 iPhone，iPad 和 Android(4.0 以上) 和 Mac。而且使用者可隨時切換成全螢幕播放，用以達到最佳的觀看體驗，不管是在行動裝置或是電腦上播放。

本篇論文需要架設網頁伺服器，而此程式設計最大好處為可使用任何一種現今常用的網頁伺服器。本篇論文使用在 Linux 下最常見的 Apache 網頁伺服器，且需要安裝 PHP 的外掛模組，網頁伺服器需要支援 PHP 網頁。實作程式是架設於 PHP

架構上，利用 PHP 撰寫跟客戶端溝通的網頁程式，PHP 為一種利用描述語言撰寫可跑在伺服端的程式。

3.2.2 系統流程

本論文主要實作兩個 PHP 程式，其特性為執行於伺服端，所以對客戶端負載不會有影響，而且只有當每次客戶端需要使用時，才會執行 PHP 程式一次。第一個 PHP 程式用於動態產生 M3U8 檔；當客戶端要求下載串流檔時，第二個 PHP 程式用於傳輸串流檔給客戶端，而且當傳輸完串流檔，即可得知傳輸時間且計算出現在頻寬。把頻寬參數代入 FFmpeg，用以設定下個串流切片檔的串流品質。

第一版程式運作流程為使用者經由程式播放介面 (如圖 3.3)，選擇出想觀看的影片後，客戶端會試著跟伺服端取得新的 M3U8 播放清單。當客戶端取得 M3U8 檔時，即可知道該如何下載串流檔，即會開始嘗試下載串流檔。客戶端下載串流檔後，即可開始播放影片，之後客戶端會一直重覆取得串流檔且播放串流檔。

原生 HLS 的 M3U8 播放清單會預先產生，HTML5 只需讀取載入 M3U8 播放清單即可 (如圖 3.4)，適用於隨選視訊環境，由於本論文需要達到即時線上串流功能性，需要支援 HLS 的線上串流格式，每個串流檔下載前，載入的播放清單皆不相同，所以實現每次客戶端請求 M3U8 檔時，可動態調整 M3U8 檔的序列號 (SEQUENE) 且儲存於資料庫中。

SEQUENE 用於記錄客戶端播放至第幾個串流檔，且令客戶端判斷是否有新的串流檔可下載且播放，而回應給各個連線客戶端的 M3U8 檔內容都是獨立不相同的，回傳的檔名也會由客戶端實體 IP 命名而成，經由 PHP 實現即時產生播放清單，當客戶端經由 HTML5 請求播放清單時，實際上為執行 PHP 程式，PHP 程式產生出播放清單且回傳給客戶端 (如圖 3.5)，圖 3.6為 PHP 程式如何動態產生 M3U8 檔的流程圖。

當客戶端取得 M3U8 播放清單後，會嘗試去播放播放清單內設定的串流檔，而程式同樣利用 PHP 當作和客戶端溝通的媒介。當客戶端要求取得可下載的串流檔時，其實是執行一次用以取得串流檔的 PHP，PHP 會先去跟伺服端下載可播放的串流檔，傳輸給客戶端，且測量出傳輸給客戶端的上傳時間，用以計算出伺服端和客戶端的頻寬變化。當取得頻寬變化量，經由公式 3.1計算出下次串流檔應該調

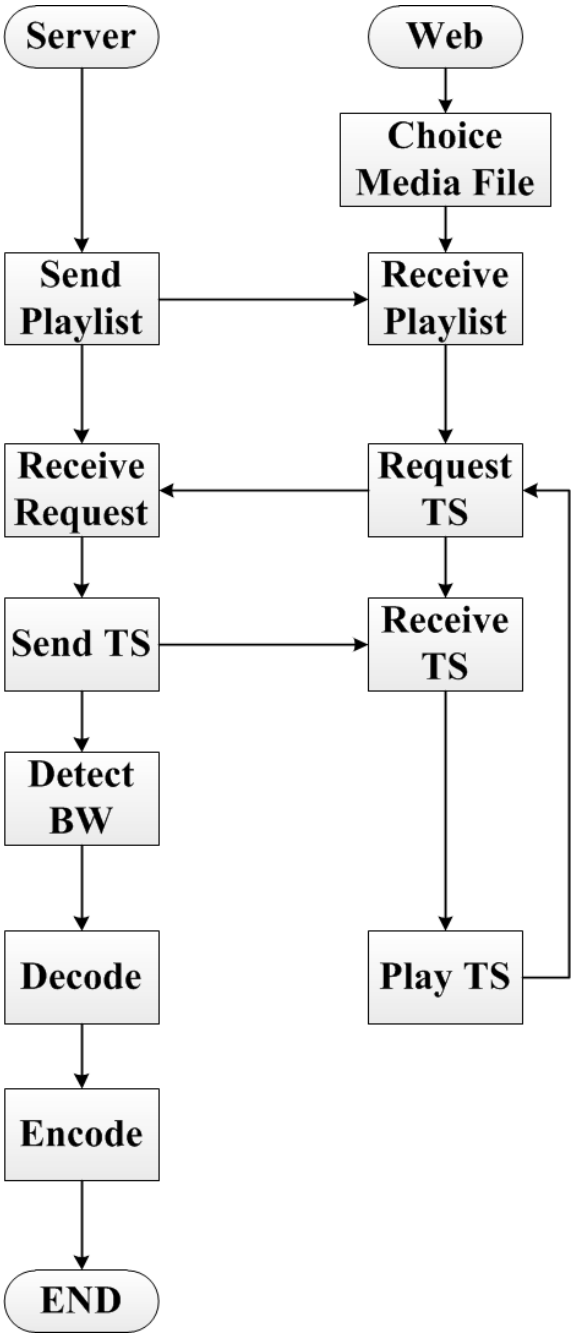

圖 3.3: 第一版程式流程圖

```
<html>
<body>
<video src="http://ip/hls.m3u8" controls autoplay>
</video>
</body>
</html>
```

圖 3.4: 標準 HLS M3U8 播放清單

```
<html>
<body>
<video src="http://genm3u8.php" controls autoplay>
</video>
</body>
</html>
```

圖 3.5: 動態產生 M3U8 檔程式碼

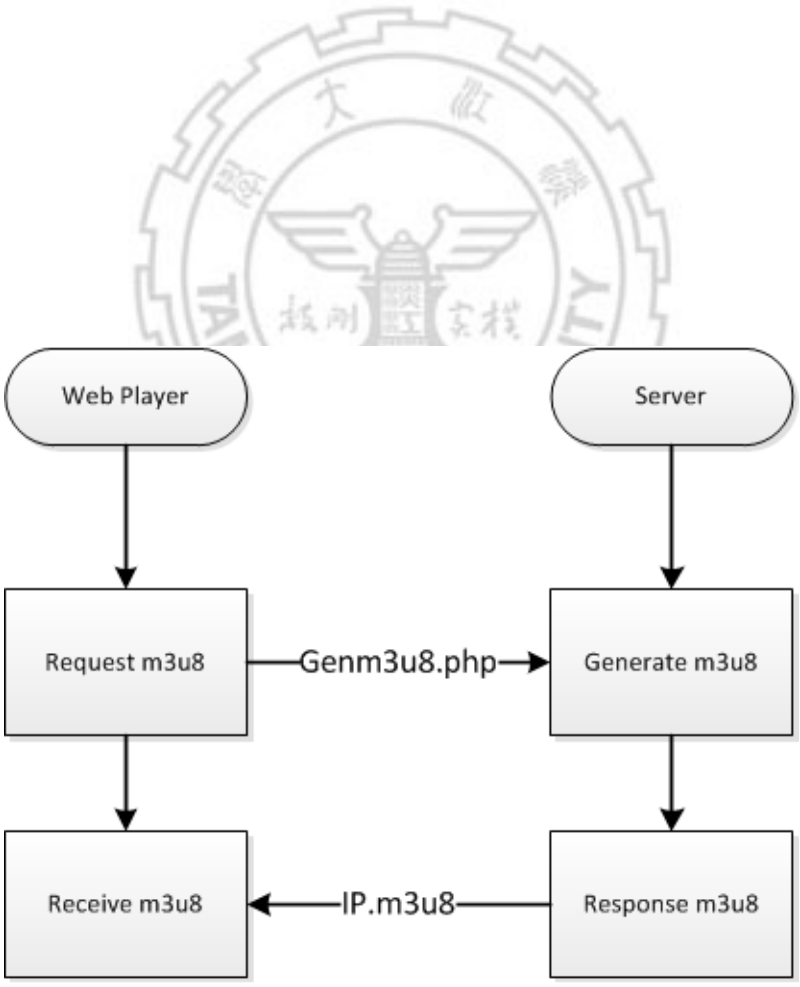

圖 3.6: 動態產生 M3U8 流程圖

整的位元率，伺服端即可開始動態編碼下一個串流檔，當作下一次被呼叫到時，傳輸給客戶端的暫存線上串流檔 (如圖 3.3)。

3.2.3 FFmpeg

FFmpeg 是個相當強的程式與函式庫。FFmpeg 是一個自由軟體，可以執行音訊和視訊多種格式的錄影、轉檔、串流功能，支援相當大量的現今影音播放格式，播放格式之間可以互相轉換，為其功能強大的特性；以及 libavformat，此為一種音訊與視訊格式轉換函式庫，提供用於 muxer 和 demuxer 等功能，可以很方便的把聲音和影像合併在一起或是各別取用；libswscale 是一種可以很方便地對於影像作縮放的函式庫。

3.2.4 動態編碼

本篇論文實作即時的動態編碼，經由 FFmpeg 完成 H.264 的 MPEG4-TS 檔編碼，快速切割出想要的影像片段，用以提供給 HLS 播放線上串流。FFmpeg 可調整的參數相當繁多，本程式將只利用其參數動態編碼把影像轉為 x264 格式的 TS，聲音將會轉為 mp3 格式，且直接指定影像輸出位元率的大小，FFmpeg 會動態調整影像的色度 (chroma)，GOP(Group of picture) 的設定和 DCT 的矩陣大小等 H.264 的相關參數。經由’-async’參數，用以強制指定每個串流檔一開始會進行影音同步，確保聲音和影像在編碼後不會發生聲音和影像無法配合，影響觀看的體驗。

3.2.5 影片字幕

視訊串流有時會觀賞些外語影片，因為語言隔閡影響觀看者對影片的理解，對使用者的觀看體驗將會降低。如果視訊串流能支援字幕功能，將可提供翻譯串流內容等資訊，也可提供串流提供者，加入一些需要提供給使用者的文字資訊，例如字幕廣告、跑馬燈字幕，用以提供各種文字資訊。實作方法將會使用 WebVTT 技術 (如章節 2.7.3)，此技術為 W3C 制定用來解決 HTML5 串流的字幕問題，本篇論文將會修改 W3C 所制定的規格，用以合乎本論文系統架構。

提供字幕功能，需要修改 M3U8 檔的輸出格式，所以需要修改產生 M3U8 檔的 PHP 程式，加入字幕的標籤 (如圖 3.7)，#EXT-X-MEDIA:TYPE=SUBTITLES

在 HLS 用以設定且提供字幕功能；GROUP-ID="subs" 用以和相對應的串流檔配對；NAME=" 日本人" 可設定顯示在播放器中，用以選擇字幕的說明文字；AUTOSELECT=YES 用以設定是否會自動選擇；FORCED=NO 用以設定預設是否會掛載此字幕；URI="subtitles.m3u8" 用以設定應該掛載那個字幕的 M3U8 檔。

```
#EXT- X- MEDIA:TYPE=SUBTITLES,GROUP- ID="subs",NAME="國語",AUTOSELECT=YES,FORCED=YES,URI="gensubm3u8.php“
```

圖 3.7: HLS 字幕標籤

字幕實作將會修改 URI 標籤中的內容，不掛載 WebVTT 字幕檔且修改為掛載 PHP 程式，此 PHP 程式將會自動產生與播放串流相對應的字幕 M3U8 檔 (如圖 3.8)，串流即可順利讀取到配對的字幕檔 (WebVTT 檔)，其程式將會讀取字幕時間軸長度，用以設定 DURATION 數值為字幕時間軸長度，動態設定 DURATION 值即可達成只需掛載單一字幕檔。標準 WebVTT 需要為每個串流影片檔皆提供一個字幕檔，其方法可讓串流提供者在串流播放中，動態更新顯示字幕，在本篇論文實作可不需達成此目的，且此方法會影響字幕掛載的複雜度。

```
printf("#EXTM3U\n");
printf("#EXT-X-MEDIA-SEQUENCE:%d\n", $seq);
printf("#EXT-X-TARGETDURATION:30\n");
printf("#EXTINF:30,\n");
printf("subtitles.webvtt\n");
```

圖 3.8: 動態產生字幕程式

3.2.6 偵測頻寬

測量頻寬在本篇論文是個很重要的課題，測量伺服端和客戶端之間頻寬，如果能越準確，對動態編碼影片品質正確性將大為提升。有效利用現有網路頻寬，同時提供最高影片品質。本章節提出了三種測量頻寬的方法。

1. 利用 JavaScript 在網頁介面撰寫測速下載程式，於每次測試頻寬時，會從伺服端下載一個測試檔案，用於測試現在頻寬。但會產生一個問題，在測試時會額外佔用現有頻寬，線上串流可使用頻寬會變少且影響播放的流暢度，而且無法準確測量出客戶端和伺服端真實頻寬變化。

2. 修改網頁伺服器的程式碼，於伺服器建立 Socket，傳輸影片串流時測量傳輸時間，用以計算出頻寬變化。但會產生一個問題，架設伺服器時，無法自由選擇適合的網頁伺服器，也難以修改全部市面上常見網頁伺服器的程式碼。

3. 最佳找出一個可解決以上問題的解法，利用 PHP 撰寫程式當作伺服端和客戶端的下載的中間媒介。當客戶端要求下載時，實際上是執行 PHP 程式，所以可輕易取得客戶端何時要求下載，可以很方便計算下載花費時間，以測量頻寬，可提供更多設計需求。

3.2.7 HLS 伺服器

HLS 當初設計時，即講求可快速架設出一個線上串流的系統。當安裝好網頁伺服器後，只需調整伺服端的 MIME 設定，用以支援 HLS 所需要支援的檔案副檔名。以架設平台為 Ubuntu 且安裝 Apache 網頁伺服器為例子，需要修改"/etc/apache2/mods-enabled/mime.conf"，增加幾筆支援的檔案格式。

3.2.8 資料庫設計

本程式還需要安裝資料庫軟體，用以儲存在程式運作中，所需要的各種數據，用以提供給程式完成全部功能的參考資料。本程式將利用 MySQL 架設資料庫環境，資料庫儲存可分為兩個儲存表格，一為 media_list，另一為 session。

表 3.3: media_list 表格

欄位	資料型態	說明
Name	CHAR	記錄影片檔案名稱
Duration	BIGINT	記錄影片播放長度 (秒)
BitRate	BIGINT	記錄影片原始位元率 (kbps)

media_list 表格用以儲存伺服端可播放的所有檔案清單，表格內記錄了檔案名稱 (Name)，時間長度 (Duration) 和影片原始位元率 (BitRate)，如表 3.3。當使用者開啟網頁介面時 (參照圖 3.9)，會讀取資料庫取得可播放的影片列表，使用者可以隨意選擇想要播放的影片。而且介面也提供搜尋影片功能，當影片增加越來越多時，檔案清單會過長，無法在有限的網頁介面顯示出來，使用者可利用搜尋功能，找出想播放的影片。

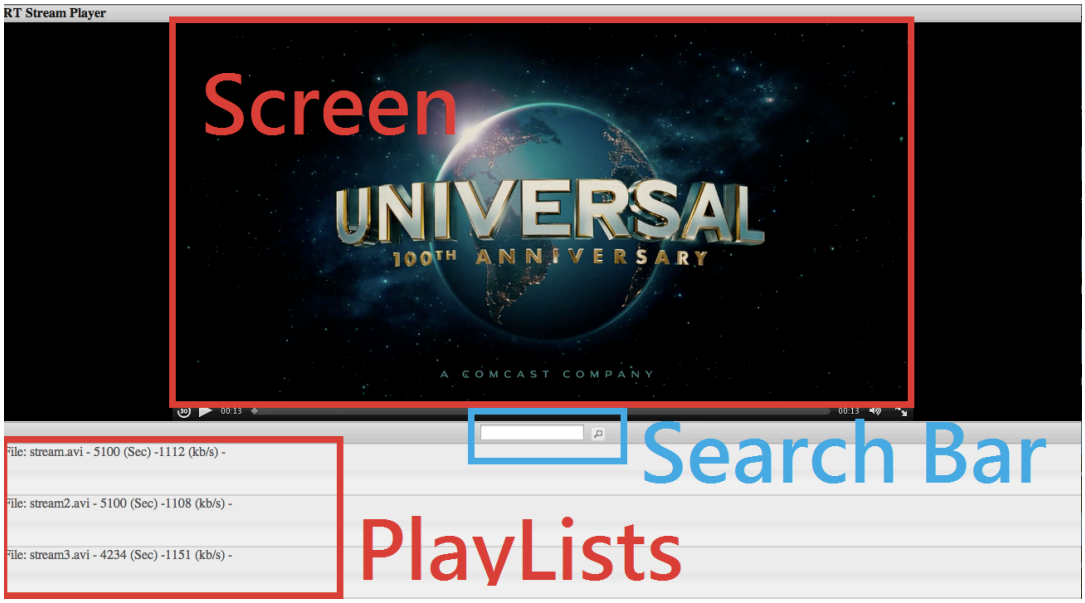

圖 3.9: 客戶端介面

Session 表格用以儲存各個使用者現在播放的狀態。表格內利用 Address 儲存了每個使用者的 IP 位址，以用區別每個使用者的儲存資料；Bandwidth 儲存每位使用者現在所測量出的頻寬變化，用以提供給下次編碼時，轉碼的參數設定；File 儲存每位使用者現在選擇播放的影片檔名，用以當作編碼時，編碼程式輸入檔名的參數設定；Sequence 儲存每位使用者現在播放進度，可和 File 合併使用，當使用者不小心關閉視窗時，可利用此參數回復狀態，知道使用者上次播放檔案進度。

3.2.9 介面設計

本章節利用 jQuery 撰寫 Web UI，其為市面上其中一種 JavaScript 函式庫，用於快速方便的設計出所需得功能，然後經由資料庫取得可播放影片的各種資料，顯示於播放清單，例如影片名稱、影片原始位元率和影片時間長度。希望提供 Web UI 給使用者操作網頁型播放程式，跟資料庫查找伺服器內可播放影片，讓使用者可選擇喜歡影片播放，提供在行動裝置和電腦上相同的操作體驗，如圖 3.10。

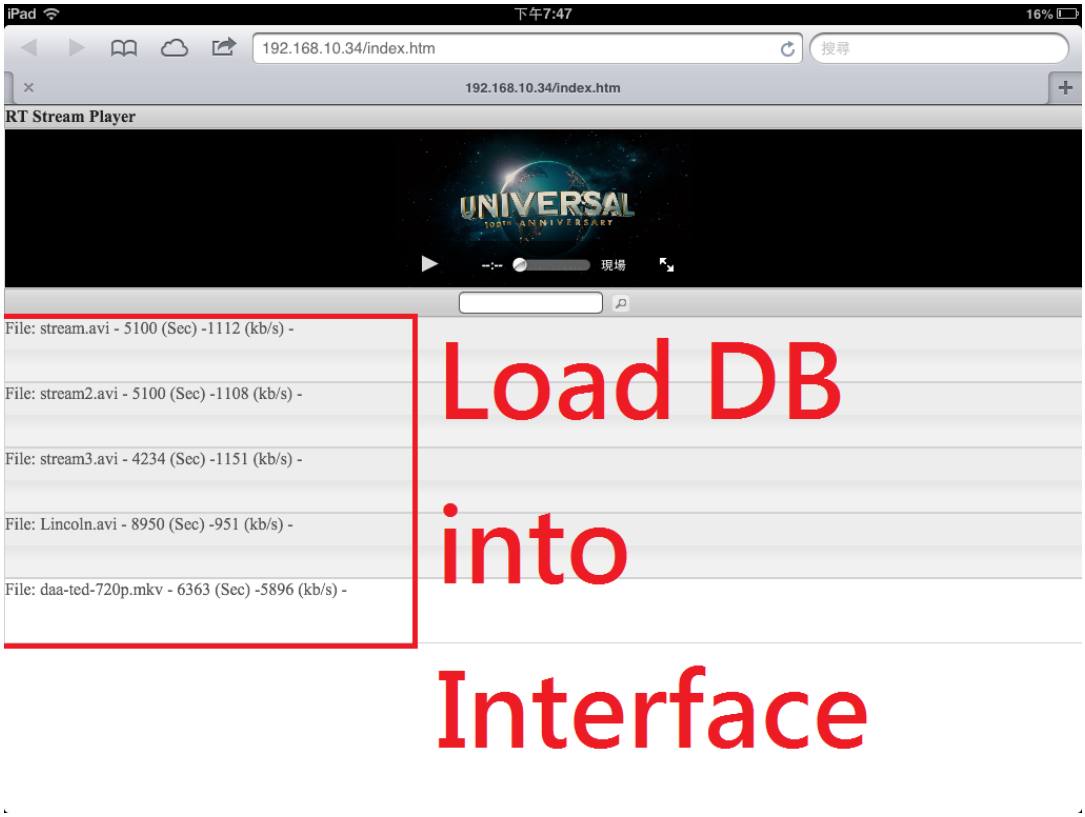

圖 3.10: 讀取資料庫資料顯示於介面上

第四章　功能分析與效能比較

本篇論文設計 AHLS 前，設立研究目標為：

1. 提供使用者不需額外安裝播放程式，只需要經由瀏覽器即可播放視訊串流，用以達成簡化使用者播放串流的使用方式。早期播放串流需要針對伺服器編碼和傳輸協定，使用相對應的播放程式，方可進行接收串流且播放影片，例如 RTP/RTSP 協定。
2. 使用者可經由行動終端設備和電腦終端設備播放視訊串流，提供使用者在各個有網路的地方，都可順利播放視訊串流。
3. 提供使用者可在各種網路頻寬下，包含家用網路、行動網路和無線網路，皆可流暢播放視訊串流。傳統串流皆會傳輸相同品質的視訊串流，其會導致當頻寬不足時，將會無法流暢播放串流，播放程式會一直緩衝其影像串流，當播放程式確定緩衝的串流，已經可以流暢且完整播放完影片，才會停止緩衝，其對使用者操作體驗將會很差。
4. 可於視訊串流中外掛字幕，串流提供者可經由字幕功能，提供視訊的翻譯字幕功能和即時文字資訊。

表 4.1: 各串流技術比較

串流技術	字幕	播放平台	外掛程式
Adobe HDS	不支援	IE、Safari、Chrome 等電腦瀏覽器	Flash
MS Smooth	不支援	IE、Safari、Chrome 等電腦瀏覽器	SliverLight
DASH	不支援	IE、Safari、Chrome 等電腦瀏覽器	DASH VLC plugin
Apple HLS	只支援隨選視訊	iOS、Mac Safari、Android	不需安裝
mp4	支援	行動裝置和電腦裝置	不需安裝
WebM	支援	Chrome、Android 2.3.3 以上	不需安裝
AHLS	支援	iOS、Mac Safari、Android 4.0 以上	不需安裝

表 4.1針對字幕、播放平台和瀏覽器需要額外安裝外掛程式進行比較。目前經由瀏覽器可順利掛載 WebVTT 字幕的平台方面，於 W3C 規範上，目前只支援 HTTP 漸近式下載協定，所以 mp4 和 WebM 可順利經由 HTML5 掛載字幕。而 HLS 和 AHLS 在字幕支援方面，由於 HLS 和 AHLS 於 iOS 平台上支援 WebVTT 字幕功能，所以字幕可於 iOS 平台上順利掛載字幕。而 HLS 只可支援於隨選視訊

上使用字幕，AHLS 可順利掛載於即時串流。外掛程式方面，目前自適式串流技術只有 HLS 和 AHLS 可完全支援 HTML5 的瀏覽器中串流播放，其他漸近式串流皆需要額外安裝外掛程式，不然無法於瀏覽器播放。而漸近式下載技術中，mp4 和 WebM 都可經由 HTML5 播放串流。

表 4.2和表 4.3將針對 HLS 和 AHLS 效能比較分析。HLS 設計在播放串流中，會開始下載下一個串流切片檔案，如果串流切片時間設定為 10 秒，瀏覽器將會在播放串流切片於 5 秒時，開始下載下一個串流切片檔案。由表 4.2分析顯示，由於 HLS 其設計位元率是固定，所以串流切片檔案將會固定大小，當頻寬越來越小時，所需傳輸時間越來越長，當頻寬 0.5Mbps 時，已經無法在播放串流時，於有限時間 5 秒內傳輸完成，將會導致視訊串流開始停頓，視訊串流無法流暢播放，將會影響使用者觀看體驗不佳。由表 4.3分析顯示，AHLS 設計會依照當前頻寬變化，自適式轉碼下一個串流切片檔的位元率，就算頻寬越來越小，AHLS 會調整位元率，使得串流切片檔案縮小，用以提升傳輸時間。在測試環境條件下，皆可於有限時間內下載完串流切片檔，視訊串流將可流暢播放，用以提升使用者觀看體驗，串流播放流暢而影像品質有些損失是可以接受。AHLS 設計是可以適應於各種頻寬變化，使用者不必擔心播放環境是否足夠播放串流。

表 4.2: HLS 測試時間

頻寬 (Mbps)	2	1	0.5	0.25
串流切片檔案大小 (Mbytes)	4.1	4.1	4.1	4.1
傳輸時間 (秒)	2.05	4.1	8.2	16.4

表 4.3: AHLS 測試時間

頻寬 (Mbps)	2	1	0.5	0.25
串流切片檔案大小 (Mbytes)	5.688	3.535	1.247	0.803
傳輸時間 (秒)	2.844	3.535	2.493	3.229

第五章　結論與未來研究

本篇論文目標為希望能依據現有頻寬，動態調整影片位元率，且在可限制的影像編碼參數，提供最佳畫質的線上串流影像，經過前面章節討論後，可得到以下結論。

1. 經由測試結果已得知，影片位元率確實可依據客戶端與伺服端之間頻寬，動態調整編碼時位元率，可讓使用者能順利在各種不同網路環境 (有線網路、無線網路和行動網路)，順利播放線上串流影片。
2. 實作出得網頁操作介面，確實可適用於 iPhone、iPad、Android 4.0 手機和 Mac 電腦等，都可提供相同的操作介面，且可流暢播放，成功達成去除客戶端程式，且去除使用 Adobe Flash 等外掛程式於瀏覽器播放。
3. 更可以擴充應用範圍，可利用平板連接電視或投影機，即可達到隨身電影院的應用功能。

在取得現在頻寬後，如何有效調整位元率為一種很重要的課題，利用現有公式可處理大部份頻寬變化情況，希望在未來可提出預測頻寬變化機制，當在取得頻寬之前，就可知道自適式編碼應該使用位元率來調整影片，除了可不必受制於等待頻寬偵測的機制，可更接近和影片播放時間同步的即時編碼，且可以更有效的調整位元率，更加不容易受到頻寬變化量大的網路環境所影響。

由於目前只支援 HLS 播放，使得可播放平台受限制，程式當初設計有考慮未來跨平台的通用性，希望未來可隨時增加支援的平台，而可達到完全跨平台的線上串流系統。

參考文獻

[1] D. Wu, Y. T. Hou, W. Zhu, Y.-Q. Zhang, and J. M. Peha, ``Streaming video over the internet: approaches and directions," in Circuits and Systems for Video Technology, IEEE Transactions on, vol. 11, 2001, pp. 282--300.

[2] Report: Video accounts for half of all mobile traffic; android biggest for mo- bile ads. [Online]. Available: http://techcrunch.com/2012/02/22/report-video-accounts-for-half-of-all-mobile-traffic-android-biggest-for-mobile-ads/

[3] T. Lohmar, T. Einarsson, P. Fröjdh, F. Gabin, and M. Kampmann, ``Dynamic adaptive http streaming of live content," in World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a, 2011, pp. 1--8.

[4] 寬頻上網評量計畫執行摘要. [Online]. Available: http://www.ttc.org.tw/docs/%E5%AF%AC%E9%A0%BB%E4%B8%8A%E7%B6%B2%E8%A9%95%E9%87%8F%E8%A8%88%E7%95%AB%E5%9F%B7%E8%A1%8C%E6%91%98%E8%A6%81-20130128.pdf

[5] 洪徹易, ``基於 http live streaming 技術之實況廣播暨 vod 系統," 清華大學資訊工程學系學位論文, 2012.

[6] 謝勝凱, ``基於 http live streaming 技術之可調式影片演算法設計與研究," 東華大學, 2012.

[7] Video on demand. [Online]. Available: http://en.wikipedia.org/wiki/Video_on_demand

[8] D. H. Finstad, H. K. Stensland, H. Espeland, and P. Halvorsen, ``Improved multirate video encoding," in Multimedia (ISM), 2011 IEEE International Symposium on, 2011, pp. 293--300.

[9] Streaming media. [Online]. Available: http://en.wikipedia.org/wiki/Live_streaming

[10] Y. Luo, J. Wang, H. Deng, and B. Hu, ``Unicast and multicast combination of unified bandwidth allocation model on cable network," in Signal Processing, 2008. ICSP 2008. 9th International Conference on, 2008, pp. 2905--2908.

[11] Live television. [Online]. Available: http://en.wikipedia.org/wiki/Live_television

[12] V. Swaminathan and S. Wei, ``Low latency live video streaming using http chunked encoding," in Multimedia Signal Processing (MMSP), 2011 IEEE 13th International Workshop on, 2011, pp. 1--6.

[13] H. Schulzrinne, A. Rao, and R. Lanphier. Real time streaming protocol (rtsp). [Online]. Available: http://tools.ietf.org/html/rfc2326

[14] Http dynamic streaming. [Online]. Available: http://www.adobe.com/products/hds-dynamic-streaming.html

[15] Smooth streaming. [Online]. Available: http://www.iis.net/downloads/microsoft/smooth-streaming

[16] E. R. Pantos. Http live streaming(ietf). [Online]. Available: http://tools.ietf.org/html/draft-pantos-http-live-streaming-11

[17] Dynamic adaptive streaming over http. [Online]. Available: http://en.wikipedia.org/wiki/Dynamic_Adaptive_Streaming_over_HTTP

[18] Mpeg. [Online]. Available: http://en.wikipedia.org/wiki/Moving_Picture_Experts_Group

[19] Mpeg-1. [Online]. Available: http://en.wikipedia.org/wiki/MPEG-1

[20] Mpeg2-ps. [Online]. Available: http://en.wikipedia.org/wiki/MPEG_program_stream

[21] Mpeg2-ts. [Online]. Available: http://neuron2.net/library/mpeg2/iso13818-1.pdf

[22] Html5. [Online]. Available: http://www.w3.org/TR/html51/

[23] Html5 differences from html4. [Online]. Available: http://www.w3.org/TR/html5-diff/

[24] Webvtt: The web video text tracks format. [Online]. Available: http://dev.w3.org/html5/webvtt/

[25] Video subtitling and webvtt. [Online]. Available: http://html5doctor.com/video-subtitling-and-webvtt/

[26] Http live streaming examples. [Online]. Available: https://developer.apple.com/resources/http-streaming/examples/

[27] Apple http live streaming. [Online]. Available: https://developer.apple.com/resources/http-streaming/

[28] K. J. Ma and R. Bartos, ``Http live streaming bandwidth management using intelligent segment selection," in Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE, 2011, pp. 1--5.

[29] K. Lai and M. Baker, ``Measuring bandwidth," in INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, vol. 1, 1999, pp. 235--245.

[30] V. Jacobson. (1997) pathchar --- a tool to infer characteristics of internet paths. [Online]. Available: ftp://ftp.kfki.hu/pub/packages/security/COAST/netutils/pathchar/msri-talk.pdf

[31] T. Kupka, P. Halvorsen, and C. Griwodz, ``An evaluation of live adaptive http segment streaming request strategies," in Local Computer Networks (LCN), 2011 IEEE 36th Conference on, 2011, pp. 604--612.

附錄 -英文論文

Adaptive Transcoding HTTP Live Streaming System

Lain-Jinn Hwang and Chien-Yi Lu
Department of Computer Science and Information Engineering, Tamkang University
Email: 700410193@s00.tku.edu.tw

***Abstract*—The most popular Internet live streaming technique, Apple HTTP Live Stream, which causes that users suffer from the consistency in bitrate when the bandwidth between the user and the server varies. Thus, this article solves the issue mentioned previously. The method we conduct makes use of adaptive transcoding technique and bandwidth management approach in order to achieve the goad that the video stream quality varies in according to the bandwidth change. We have the stream fluency the major consideration while the maximum bandwidth utilization the minor to assure the highest video quality.**

***Keywords*—HTTP Live Streaming, Mobile Streaming, Bandwidth Detection, Adaptive HTTP Streaming, WebVTT.**

I. Introduction

WITH the fast spreading of Internet connections and increasing landlines and mobile bandwidth for data upload and download, network based multimedia services are getting diversified every day. Take network streaming for example. From the network broadcasting in the early days to the HD video streaming now, users are not using devices for page browsing only. A recent research report suggests that more than half of bursting global traffic can be attributed to network streaming [1][2]. Thanks to the popularization of smartphone and tablet computer, a full range of cloud services is now true. This adds more diversified network streaming platforms. Cloud services have shifted from limited indoor locations on landline networks in the early days to browsing anywhere you go, either indoor or outdoor [3]. Ever the larger screen size and higher resolution of smartphone and tablet computer enables viewers to enjoy better quality program [3]. The Internet connection traffic by smartphone in Taiwan is soaring. Latest mobile network bandwidth tests indicate that network connection in Taiwan is relatively unstable with speeds varying at different locations. The worst place for networking is on a moving MRT. Time is another factor in determining communication speed with smartphone. Average phone connection speed is relatively better from night to before working hours and worse from working hours to one o'clock in the morning. Despite the adoption of LTE high speed network, the installation is still in process and people in Taiwan may still suffer limited bandwidth. Before the mobile network system is well set up in Taiwan, browsing the Internet or watching contents streamed by it may leave you poor experiences. Outdoor networking bandwidth varies more significantly than in an indoor environment. The strength of local signal, number of users connected to the base station, local bandwidth and changes in bandwidth caused by movement all may impact online streaming. Any environment with unstable bandwidth is worth studying, ensuring users' optimum viewing quality by balancing screen definition and fluency [4][5].

Conventional network streaming mandates the installation of the same program in both server and client end computers. The server end compresses contents with fixed bit rate (image quality) and the same encoding format and sends both video and audio contents to the client end through TCP, UDP or RTP/RTCP protocols. The client end computer then decodes and plays back received streaming contents. With fixed bit rate compression and pre-encoding, it can easily provide high definition video in case of sufficient bandwidth. However, when playing this kind of video in a network with limited bandwidth, viewers suffer from continuous buffering status. As more people are connected through mobile network and live in a fast moving mode, they require fast information retrieval and instant services rather than waiting for the download of films. This study tries to solve this problem by providing users with instant viewable HD streams without encoding videos in a supported format in advance.

A lot of network platforms require online stream viewers to install player software that supports the desired streaming services before viewing contents offered by them. Users are required to select a player program but not all of them have this capability. A better way of selecting the proper program to view streams should be provided instead of asking users for player selection on the basis of format of contents to be played. User experiences can be improved a lot if their operation can be simplified if required services can be accessed by opening their browsers.

II. Background

The HTTP Adaptive Streaming (HAS) technology combines the features of conventional streaming technology and HTTP's progressive download and playback to deliver media contents in HTTP. The HAS improves users' media playback experiences while reducing technology complexity at the server end. This makes it adopted as the trend in video streaming development.

HAS technology integrates features of conventional RTSP/RTP streaming media technology and HTTP's progressive download to be of high efficiency, expansion, and compatibility.

The HAS technology is a mixed media transmission. The contents may look like being streamed to viewers. In reality, it delivers contents by employing HTTP protocol's progressive downloading. Media contents are divided into a series of media blocks for pull-based transmission. That is, each media block is transmitted only at the request by the client end computer. Its core technology lies in slicing videos into fixed play time

block, usually 2-10 seconds. At the video encoding layer, this implies that every slice is composed of certain number of complete video GOPs, i.e., every slice contains one key frame, to ensure independence of every slice from its last and next counterpart.

Media slices are saved in HTTP Web server. The client requests slices from the server in linear manner and download them in HTTP. Once the media slices arrived at the client they are played back in sequence. As streamed slices are encoded in given conventions there is no lost or duplicated content. This ensures viewers seamless and smooth playback of streaming contents.

In case a copy of the contents are encoded with multiple bit rates, then the contents slicing module can cut it into slices of individual bit rates. As the Web server streams by making the most of available bandwidth without traffic control mechanism, the client then can choose to download larger or smaller media slices for bit rate self adaptiveness by detecting valid bandwidth from the source Web server.

The core technology of HAS is composed of two parts. One is content preparation including the transcoding platform that supports multiple terminal devices and the media slicing module. The other is the streaming for content delivery on the basis of HTTP based media source server and the terminal equipments oriented contents delivery network.

A. Apple HTTP Live Streaming

Conventional network streaming mandates the installation of the same program in both server and client end computers. The server end compresses contents with a fixed bit rate (image quality) and the same encoding format and sends both video and audio contents to the client end through TCP, UDP or RTP/RTCP protocols. The client end computer then decodes and plays back received streaming contents. A better way of selecting the proper program to view streams should be provided in an era of cloud instead of asking users for player selection on the basis of the format of contents to be played. User experience can be improved a lot if their operation can be simplified and if required services can be accessed by opening their browsers.

Conventional RTP protocol based online streaming technology employs a special protocol and communication port and makes it easy to be blocked by firewall unless the port is exclusively opened for it. The design requires it to maintain being connected throughout the entire playback session. The server end is then required to manage every connected client with a single and exclusive streaming session. This consumes a lot of resources at the server end. On the contrary, the HAS technology requires the server end to process packets in sequence by employing HTTP protocol to deliver them. This not only reduces server end workloads but also avoids being blocked by a firewall as the communication port required by HTTP protocol is the one that must be opened for web server.

Despite enabling users to view video streaming through a browser, Adobe Flash Dynamic Streaming and Microsoft Smooth Streaming suffers from the inconvenience of plug-in installation, as neither are supported by the browser itself.

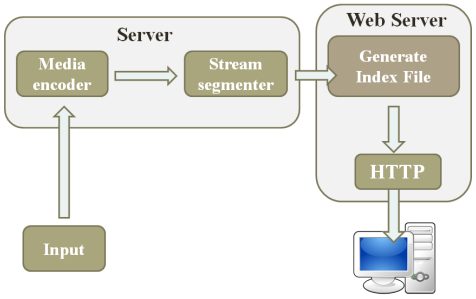

Fig. 1: HLS architecture

This is not user friendly. As mobile devices, smartphones and tablet computers are getting popular everyday and most do not support plug-in added in their browsers, most network streaming is viewed in a conventional manner where a special application is installed to keep the connection with the server for video transmission. Apple HTTP Live Streaming is few of the systems that support network stream viewing in a browser.

Differing from other RTP/RTSP online streaming technology, the HLS employs HTTP protocol rather than a customized one. This helps in avoiding to be blocked by firewall or proxy server. The HTTP protocol's TCP data transmission mechanism also ensures A/V quality and correctness. The HLS system also has merits in that it enables you to set up a VOD online streaming system with any popular web server in a fast and easy way. That is why this thesis selects HLS as the technology to implement the theory.

HTTP Live Streaming employs HTTP protocols. This comes with two benefits. The first is that video data can be delivered by a browser such that service pages can be revised without any need of software installation at the client end. The second is that it is less likely to be blocked by a router firewall. Viewers can access network streaming with browser behind routers without changing router's firewall rules.

See Figure 1 for the structure of HTTP Live Streaming. Once a video clip arrives, it is compressed and encoded into a MPEG2-TS before cutting into short and small Transport Streams of 10 seconds length, recommended by Apple Inc. A playlist of M3U8 format is then generated that contains URL of every Transport Stream. Viewers then can play back the video by opening the M3U8 playlist in their browser. Embedding the M3U8 playlist in an HTML5 page may ease the playback even further. The HTTP Live Streaming offers multiple definition formats with the M3U8 playlist. Viewers may select proper playback definition based on available bandwidth and image quality data stored in the M3U8 index file.

The basic theory is to slice one Transport Stream (TS) into many pieces of short time span streaming files. After one streaming file is downloaded, the streams starts to provide to the user. As each TS is short and small, users can start their viewing quickly with reduced streaming delay. In addition, when playing online streaming in browser, the server end delivers multiple TSs with different definitions and bit rate for the browser. The latter then selects proper definition based on

```
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXT-X-MEDIA-SEQUENCE:1
#EXTINF:10,
http://url/segment0.ts
#EXTINF:10,
http://url/segment1.ts
#EXTINF:10,
http://url/segment2.ts
#EXT-X-ENDLIST
```

Fig. 2: M3U8 Format

```
#EXTM3U
#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=200000
http://ALPHA.mycompany.com/lo/prog_index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=200000
http://BETA.mycompany.com/lo/prog_index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=500000
http://ALPHA.mycompany.com/md/prog_index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=500000
http://BETA.mycompany.com/md/prog_index.m3u8
```

Fig. 3: M3U8 index file

```
WEBVTT

00:11.000 --> 00:13.000
<v Roger Bingham>We are in New York City

00:13.000 --> 00:16.000
<v Roger Bingham>We're actually at the Lucern Hotel, just
      down the street

00:16.000 --> 00:18.000
<v Roger Bingham>from the American Museum of Natural
     History
```

Fig. 4: WebVTT subtitles

```
    <video width="640" height="480" controls>
    <source src="video.mp4" type="video/mp4" />
    <source src="vidoem.webm" type="vidoe/webm" />
    <track src="subtitles.vtt" srclang="zh" label="
     Chinese" />
    </video>
```

Fig. 5: The formats of WebVTT

available bandwidth. This helps users in maintaining playback fluency in spite of changing bandwidth. To play video or audio files from browser through HLS, an M3U8 file must be added in HTML5. This is how an online streaming can be played with M3U8 file.

B. M3U8 Playlist

To enable a browser to play TS, with format as shown in Figure 2, in sequence, the tag EXT-X-TARGETDURATION is set up each TS's play time contained in the playlist. The tag EXTINF set up length of individual TS by connecting to its URL.

HTTP Live Streaming provides multiple playback definitions with a index file, see Figure 3, The tag EXT-X-STREAM-INF sets up relevant M3U8 playlist based on available bandwidth and enables the client end to choose videos.

HTTP Live Streaming is employed by video on demand services where the film is pre-encoded and saved in the server end. It adjusts the time line to the desired time point for playback without having to wait for reading the entire file before playing it back. This reduces viewers' waiting time and improves viewing experiences. However, if the input video is a real time image, e.g., webcam, then only one output format can be generated immediately. Here the client end may not select playback definition in accordance with available bandwidth and the M3U8 index file.

C. Subtitle file (WebVTT)

With HTML5 support, a browser is now able to play videos without installing plug-ins. However, without the help of subtitles viewers may lack the ability to understand videos in a foreign language. The HTML5 video subtitle, defined in WebSRT format, exists as part of HTML5 standards to display subtitles via general SRT document. It was renamed WebVTT later with subtitle standards originated from the HTML5 one. It is a standards system now[6][7].

With extension ".vtt" and UTF-8 code, WebVTT is a plain text file, see Figure 4, which supports multi-language subtitle display. It contains two types of information. One is the timeline and the other the subtitle texts. It begins with the WEBVTT markup trailed with a carriage return symbol. The timing reminder is in the format of HH:MM:SS.sss. The beginning and ending reminder is composed on one blank space, two hyphens, and one greater than symbol and separated with another blank space. The timing reminder starts with a new line and ended with a carriage return symbol. The accompanying subtitle text follows. The subtitle texts may take one or more lines without any blank line in between. The WebVTT file's MIME type is "text/vtt".

As shown in Figure 5, the WebVTT uses track tag to insert vtt subtitle file to combine HTML5 to output subtitle texts in video streams. The time line contained in the subtitle file works together with its counterpart in the streaming file to synchronize subtitle texts with the video contents [6]. As the HTTP Live Streaming slices videos into huge amount of short time span clips, the original WebVTT does not fully applicable with HLS. The time line in a single subtitle file cannot synchronize with all video files. Here individual WebVTT subtitle file is required for every video file to display WebVTT subtitle file in HLS streamed film [8].

III. Adaptive Transcoding HTTP Live Streaming System

Video on demand system enables viewers to watch pre-recorded and processed films. It enables users to jump to a

desired position in the film at the cost of huge storage spaces for keeping all the video files provided by the server. It also leads to paused playing and poor user experience when a lack of adequate bandwidth occurs during playback.

Real-time live streaming is aimed at providing viewers with instant input signal based webcasts. It enables viewers to watch instant events, e.g., online webcasting of basketball games. The drawback of it is paused images or heavy lag time when the game is going on. Both may dampen users' viewing experiences. The most critical challenge to instant streaming is to reduce image lag time and to allow re-play during the game. This saves storage space at the server end at the cost of server's computing power for huge amount of code conversion and fluctuating number of connections. The most difficult part is its constant image output bit rate. This may halt the entire streaming when there is a lack of bandwidth.

Both approaches lead to similar difficulties. That is, the playback fluency is vulnerable to changing bandwidth and number of active concurrent viewers. This paper aims to give a model that fits current network environment and capable of dealing with conventional issued described herein.

Standard HLS fits the role of video on demand services. To enable a browser to support a dynamic regulating video bit rate the server end must prepare multiple video files of different bit rates for the same film and set up conditions in the M3U8 file to guide the browser to select, download, and play proper video file based on available bandwidth such that optimum video quality can be reached [9]. An M3U8 file that lacks bandwidth criteria may select proper video quality on the basis of browser's bandwidth regulation algorithm. As this algorithm is a "greedy" one it tends to result in an undesired outcome. Whenever a new connection is established, the "greedy" algorithm lets each connection get the maximum definition settings. If the server end bandwidth fails to let every connection keep its current image quality, every browser will select to set its image quality to the minimum level provided by existing streaming. At the next playback interval (10 seconds), every connection upgrade its quality to the next higher level. If existing total bandwidth still fails to meet the needs, all the connections go back to the bottom level again or stay at the level and wait for the next 10 seconds to upgrade to the next higher level. This process of test and setup may go on and affect the viewing quality experienced by every connected user. [10] tries to solve this problem by adding one bandwidth regulating server in between the server and client ends. When a new connection is established, the bandwidth regulating server identifies every connection's best video quality at existing bandwidth then starts its playback service. When the newly added connection starts playing, all other connections are in their best image quality setting and have the issue solved.

However, [10] leads to some new issues as well. The first is that is applies to VOD services only as only the VOD system can provide all its videos' image quality settings for the bandwidth regulating server to derive the best video quality of each connection. The second is that the bandwidth detection method is not presented in detail. Two types of bandwidth must be taken into account. The one is the total bandwidth provided by the server and the other the maximum bandwidth of both the server and clients or the maximum bandwidth for the clients' networking. If the optimum bandwidth given by the bandwidth regulating server outruns the maximum connection bandwidth available at the client end, it may still delay streaming for the clients' playback.

This paper tries to solve the problem herein with the Adaptive Transcoding HTTP Live Streaming System (AHLS) with its theoretical foundation and practices described in the following.

A. Bandwidth detection

HTTP adaptive streaming comes with two key points: To provide optimized image quality and reduce lag time from image resources arrived at the server throughout to the client end for playback. This is an end-to-end(e2e) delay[11]. The standard HLS design requires to slice video files into fixed length streaming files before sending to the client. This differs from RTSP structure's compression-when-streaming approach and so is doomed of certain e2e delay. The key is to detect bandwidth without lengthening e2e delay. This paper tries to add one virtual middle layer (VML) between the server and client end to replace the connection between them. When a client requests to download HLS streaming files, it is the VML that connects with and transmit streaming to the client and to find out the transmission time of each streaming slice. With a given size of each streaming slice ε_s and streaming transmission time τ_0, we can get the existing bandwidth β between server and client ends with formula 1.

$$\beta = \frac{\varepsilon_s}{\tau_0}, \tag{1}$$

B. Dynamic trascoding algorithm

The standard HLS structure cuts streaming into many short video streaming files. In case of fluctuating network bandwidth, it changes image quality at the next slice to regulate the streaming bit rate. Based on method proposed in this paper, we can find out existing bandwidth through bandwidth detection to compress and slice streaming output file at the server end instantly. As every connection's current bandwidth is known, it can be used as the key parameter for streaming compression to ensure the streaming file bit rate at next time point would not exceed available bandwidth.

C. The encoding compression model

The existing bandwidth can be measured by recording streaming file's download time. When the client end starts playing back the streaming file, we can use the idle time, and the server end may use the measured bandwidth for next streaming file's compression parameter.

The transcoder has two functions. The one is to encode and the other decode. For source video not in format of HLS's genuine supported MPEG2-TS or MPEG4-TS (H.264), it must be decoded into non-compressed format before being encoded into streaming file. This chapter tries to give two transcoder prototypes. The HLS is designed to encode one TS file for

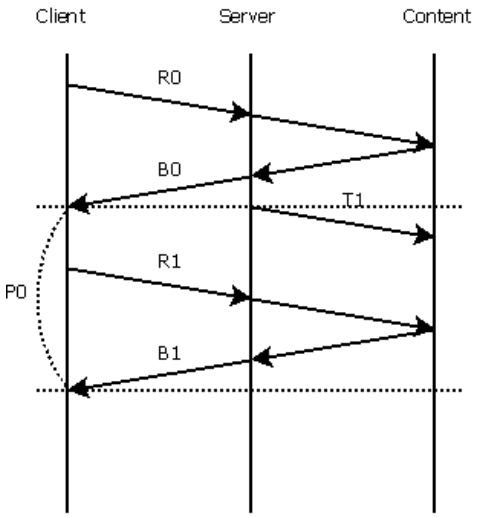

Fig. 6: Encoding model(first version)

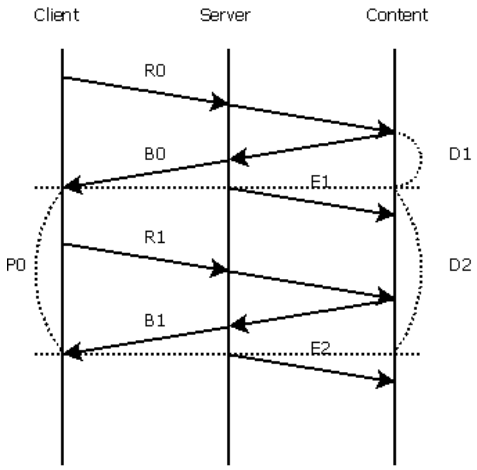

Fig. 7: Encoding model(second version)

TABLE I: The parameter of encoding model(first version)

R_0	Request first slice
B_0	Response first slice and detect bandwidth
T_1	Using bandwidth to request transcoding
P_0	Play slice
R_1	Request next slice
B_1	Response next slice and detect bandwidth

TABLE II: The parameter of encoding model(second version)

Parameter	Description
R_0	Request first slice
B_0	Response first slice and detect bandwidth
D_1	Decode first slice before
E_1	Encode first slice
P_0	Play slice
R_1	Request next slice
B_1	Response next slice and detect bandwidth
D_2	Decode second slice before
E_2	Encode second slice

every ten seconds and send the next TS file to the client when the previous one is being played back.

When sending video file to the client end, the model measures changes in bandwidth based on the download time. Figure 6 and Table I indicates that every streaming file starts to be compressed only after the previous transmission is completed. The browser requests the next streaming file when last TS file has been played for half of the time (5 seconds). The time available for encoding next streaming file is also half of the video length (5 seconds). This may lead to a problem. That is, video of higher quality may be unable to be processed here if the time required for compression is greater than 5 seconds. In case the server failed to compress next streaming file in limited amount of time, subsequent streaming download and playback may be delayed and the client end may wait for another queue time (10 seconds) before requesting next streaming file download. This may result in delayed viewing and poor viewing experience.

The second transcoder prototype tries to overcome this with faster transcoding. As shown in Figure 7 and Table II, the second prototype split the function of a transcoder into encode and decode. The server starts decoding the next streaming file before the last streaming file has been downloaded successfully. As decoding does not wait for the completion of bandwidth detection, the server end starts encoding next streaming file when the client end finished downloading the streaming file and detected existing bandwidth as well as starts playing the streaming file. This differs from the first protocol in giving ten seconds for the transcoder's necessary operation as only encoding into the streaming file is required between the bandwidth is measured and the client requests for the next streaming file.

IV. Implementation

With the popularization of mobile devices, users are getting used to browsing the Internet and watching streaming contents by connecting to mobile or wireless networks. However, mobile network's transmission speed is subject to many external factors that may lead to unstable streaming and random pauses which, in turn, dampens viewers' experience and cannot meet the fast video viewing.

To watch online streaming with mobile devices requires the installation of players to connect to the server for playback in most cases. Wouldn't it better and more convenient for viewers to play films with a browser? This helps streaming service provider to focus at web page coding and streaming service provision instead of worrying about platform compatibility and App coding.

In this section I'll outline, in directions addressed below, the software to be used, how to apply them in this paper's practice as well as potential issues, and how to solve them for

each practice presented here. I'll try to improve all the issues encountered to give designs that meet users' requirements.

A. System environment

The system environment is set up as illustrated below:

1) Install web server that supports PHP. Take this practice. It installs Apache Web Server along with PHP module.
2) Set up database system and create relevant tables. Take this practice. It installs the MySQL database in the system.
3) Copy practice program code to the default directory in web server

Once all the installations are completed, the file list database is updated and the system is ready to provide users with desired videos for viewing by saving videos to be streamed in the video folder. A cross platform online streaming player system is made by author of this paper for playback on mobile and computer devices including iPhone, iPad, any Android (4.0 or later) devices and Mac. Users may switch to full screen playback at any time, either mobile devices or computer equipment, for the best viewing experiences.

A web server is required in this paper's example. The beauty of this design is that it fits with any popular web server available now. This paper employs the most popular Apache web server available with the Linux platform along with PHP plug-in to support PHP web pages. The practice program is PHP structure based. It interfaces with the client end in PHP coded web pages. PHP is a descriptive language that can code program to run at the server end.

B. System flow

This paper coded two PHP programs that run on the server end. It brings no burden on the client end and runs only when the client end requires it. The first PHP program dynamically generates an M3U8 file when the client end requests a streaming file download. The second one sends a streaming file to the client end and derives current bandwidth based on consumed transmission time after the transmission is completed. The bandwidth data then is passed into FFmpeg to set up streaming quality for the next streaming slice.

See Figure 8 for operation flow of the program (first edition). Here, the user selects his/her desired videos in the playback interface, the client end then retrieves a new M3U8 playlist from the server end, the client end then determines the way to and starts to download streaming file based on the M3U8 file received, it then starts playing the video file after the streaming file is downloaded. The client end repeats the process to stream for the viewer.

Native HLS's M3U8 playlist is generated in advance. What the HTML5 needs to do is to read and load the M3U8 playlist as shown in Figure 9. This is ideal for VOD environment but not this paper. This paper is aimed at instant online streaming and requires a format that supports HLS online streaming. Before the download of every streaming file, the loaded playlist varies. This enables sequence of the M3U8 file's dynamic adjust and storage in database every time when a M3U8 file is requested by the client end.

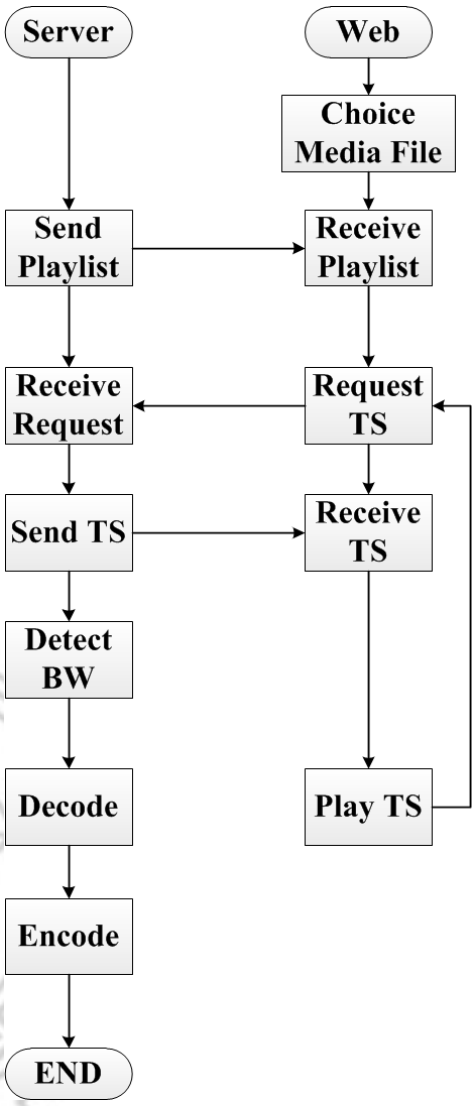

Fig. 8: Flowchar(first version)

```
<html>
<body>
<video src="http://ip/hls.m3u8" controls autoplay>
</video>
</body>
</html>
```

Fig. 9: Standard M3U8 playlists of HLS

The SEQUENE number indicates streaming file ID being played at the client end and enables it to check for new streaming files for download and playback. The contents of the M3U8 file replied to each connected client end varies from each other with its name varies with the client end's static IP.

```
<html>
<body>
<video src="http://genm3u8.php" controls autoplay>
</video>
</body>
</html>
```

Fig. 10: The code of dynamic generate M3U8

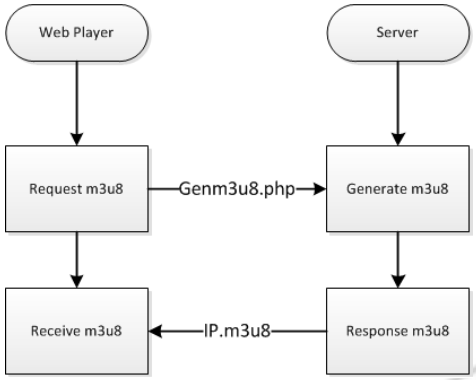

Fig. 11: The flowchart of dynamic generate M3U8

This makes instant playlist generation by PHP possible. When the client end requests for a playlist through HTML5, it runs a PHP program to generate the playlist and send it to the client end as shown in Figure 10 and Figure 11. It illustrates the M3U8 file's dynamic generation process.

After the M3U8 playlist is received at the client end, it plays the streaming file set by it and interfaces with the client end in PHP. When the client end requests to get a streaming file that can be downloaded, it runs the streaming file retrieval PHP once. Here the PHP downloads a playable streaming file from the server end, sends it to the client end, and measures the time required for transmission to check bandwidth changes between the client end and server end. It then uses the new bandwidth variance to get the new bit rate for next streaming file with formula 1. The server end then starts encoding next streaming file as the cached online streaming file for next polling as shown in Figure 8.

C. Dynamic transcoding

This paper practices a instant dynamic mechanism to encode a H.264 based MPEG4-TS file through FFmpeg, slice it into desired video clips in a fast pace, and delivers it to HLS for online streaming playback. FFmpeg features scores of parameters. This program employs its dynamic encoding parameter to convert images into x264 format based TS and sounds into mp3 format based file as well as to specify bit rate of image output. The FFmpeg dynamically adjusts relevant parameters of H.264 including images' chroma, group of picture (GOP) settings and sizes of DCT matrix. With the "-async" parameter it forces every streaming file to synchronize its audio and video

```
#EXT-X-MEDIA:TYPE=SUBTITLES,GROUP-ID="subs",NAME="Chinese
     ",AUTOSELECT=YES,FORCED=YES,URI="gensubm3u8.php
```

Fig. 12: The tag of HLS subtitles

```
printf("#EXTM3U\n");
printf("#EXT-X-MEDIA-SEQUENCE:%d\n", $seq);
printf("#EXT-X-TARGETDURATION:30\n");
printf("#EXTINF:30,\n");
printf("subtitles.webvtt\n");
```

Fig. 13: Dynamic generate subtitle

file in the beginning to ensure concurrent A/V encoding and playback for the best viewing experiences.

D. Video subtitle

For streaming file shot in a foreign language, viewers may lack the required language capability for full comprehension and may end up with poor viewing experience. Video streaming that supports subtitle function not only can provide translation texts but also can enable the streaming service provider to add some extra text information including subtitle ads. and banners. The practice of this paper employs WebVTT technology. The WebVTT technology is designed by W3C to provide subtitle for HTML5 streaming. This paper will revise the W3C specifications to comply with the proposed structure.

The output format of the M3U8 file needs be modified to provide subtitle function by changing the PHP program for M3U8 file generation with added subtitle tag as shown in Figure 12. HLS's #EXT-X-MEDIA:TYPE=SUBTITLES tag is used for set up and provide subtitle function; GROUP-ID="subs" is used to pair with corresponding streaming file; NAME="" can be set to display in player program for subtitle selection; AUTOSELECT=YES is used for auto selection ON or OFF; FORCED=NO is used for subtitle load default On or OFF; and URI="subtitles.m3u8" is used for selecting M3U8 file for mounting.

The subtitle practice modifies the contents in the URI tag from mounting the WebVTT subtitle to mounting the PHP program that auto generates subtitle M3U8 file paired with the playback streaming as shown in Figure 13. With this modification, the streaming file then reads in the paired subtitle file (WebVTT file). The program retrieves length of the subtitle's timeline to set up the DURATION parameter dynamically for single subtitle file mounting. Standard WebVTT requires individual subtitle file for each streaming video to enable streaming provider refresh subtitle during playback. This provides an easier approach while reducing the complexity in subtitle file mounting.

E. Bandwidth detection

Bandwidth measuring is critical. Precise bandwidth measuring between the client and server ends may improve quality

TABLE III: The table of media_list

item	type	description
Name	CHAR	Record video file name
Duration	BIGINT	Record video length(second)
BitRate	BIGINT	Record video bitrate(kbps)

of dynamic encoded videos and make the most of available network bandwidth. This paper provides three bandwidth measurement methods.

1) Provide speed measurement program for download at web page interface in JavaScript. Download a test file from the server end to measure current bandwidth. There is an issue with this method. The measuring activity consumes existing bandwidth that cuts bandwidth available to online streaming, worsens playback fluency, and does not measure the actual bandwidth variation between the client and server end accurately.
2) Revise program at web server to build up a socket in the server to measure transmission time during video streaming for bandwidth variation calculation. There is an issue with this method. You don't have the flexibility in selecting a proper web server when setting up the server nor the capacity to revise a program found in every market available web server.
3) The best method in solving these issues is to code a PHP program as the download interface between the client and server end. The client end requests a download by running a PHP program for easy download as required by the client end and easy measuring download time for bandwidth variation calculation. This helps in provide design requirements.

F. HLS server

The HLS is designed to set up an online streaming system in fast pace. Once the web server is set up, you can adjust server's MIME settings to support extensions required by HLS. To set up a Ubuntu platform with Apache web server installation, you need to modify "/etc/apache2/mods-enabled/mime.conf" for couple of file format supports.

G. Database design

This program requires a database system to store program operation data as the reference in completing every required function. This program set up a MySQL database with two tables named "media_list" and "session".

The "media_list" table stores list of files that can be played by the server end. This table contains fields of file name (Name), lasting time (Duration) and original video's bit rate (BitRate) as shown in Table III. Once a web interface is opened, see Figure 14, it retrieves a list of playable videos in the database for users' selection. A video query function is provided in the page. For a file list too long to display in a limited number of pages, viewers may find their desired video by this query function.

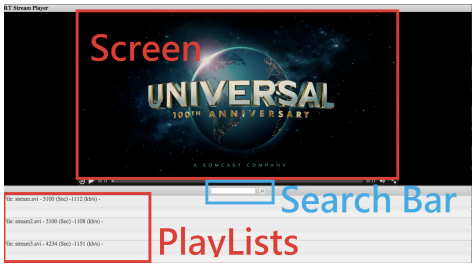

Fig. 14: Client interface

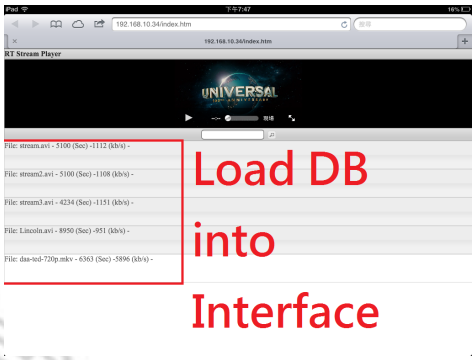

Fig. 15: Loading database into interface

The "session" table keeps current playing status of each user. Its "address' field maintains users' IP address to separate their storage data. The "bandwidth" field keeps measured bandwidth variation of each user that is used for set up the conversion parameter for next streaming file encoding. The "file" field keeps name of the file users selected for viewing. It is also the value of the file parameter for video encoding. The "sequence" field keeps users' current playback status. When used together with the "file" field it resumes the playing from where it is stopped by any interruption, e.g., user closed the program unexpectedly.

H. Interface design

This section codes a Web UI with jQuery. The latter is a commercial JavaScript library for easy and fast function designing. It retrieves data of files that can be played from the database to display in the playlist including name, original bit rate, and duration of the video. This Web UI enables users to operate web based player and query the database for videos available at the server with the same interfaces in mobile devices and computers as shown in Figure 15.

V. Performance Analysis

This paper is aimed at following goals before designing the AHLS:

1) Enables users to play streaming file in browser without installing any plug-ins for simplified streaming playback operation. Early streaming playback requires server end

TABLE IV: The different of streaming technology

Streaming	Subtitle	Environment	Plug-in
Adobe HDS	No Supported	IE, Safari, Chrome	Flash
MS Smooth	No Supported	IE, Safari, Chrome	SliverLight
DASH	No Supported	IE, Safari, Chrome	DASH VLC plugin
Apple HLS	Only Support VOD	iOS, Safari, Android	No need
mp4	Supported	All devices	No need
WebM	Supported	Chrome, Android 2.3.3 and up	No need
AHLS	Supported	iOS, Mac Safari, Android 4.0 and up	No need

coding and transmission protocol specific players to play streamed file, e.g. RTP/RTSP protocol.

2) Users may play video streaming with mobile or computer devices for streaming file viewing as long as a network connection can be established.
3) Enable users to play fluent video streaming in any networking bandwidth including landline, mobile and wireless environment. Conventional streaming service transmits files of the same quality that may impedes fluent streaming playing when lack of bandwidth. In cases like this, the player keeps on buffering streaming files until it can play the entire file fluently. This leads to poor viewing experience.
4) Enable mounting streaming in video streaming to provide translation texts for the video and other instant text information.

Table IV Reviews subtitle, playing platform, and browser with plug-in. With respect to browser that can mount WebVTT subtitles, the W3C now supports HTTP progressive download protocol and so mp4 and WebM may mount subtitles through HTML5 successfully. As HLS and AHLS is concerned, both support WebVTT subtitles on iOS platform. This enables the latter to mount subtitles successfully. HLS supports subtitles for VOD only while AHLS for instant streaming. With respect to plug-ins, current self-adaptive and progressive streaming technology mandates plug-ins for playback with a browser except HLS and AHLS. The latter two fully support streaming file playing in a browser with HTML5 support. In progressive download technologies, both mp4 and WebM can play streaming files with HTML5.

Table V and Table VI review performance of HLS and AHLS. The HLS technology is designed to start downloading next streaming slice when playing last slice. For a streaming slice of length in 10 seconds, the browser starts downloading the next slice when the playing of last slice has lasted for 5 seconds. Table V indicates that HLS's fixed bit rate design results in streaming slices in fixed size. When bandwidth is getting smaller it requires more time to transmit a file of the same size. For a bandwidth of 0.5Mbps, the streaming playback stops and the transmission cannot complete in 5 seconds. This hampers users viewing experience as the playback becomes broken. Table VI indicates that AHLS's design enables it to adjust next streaming slice's bit ratio according to current bandwidth variation adaptively. When bandwidth is getting smaller, the AHLS changes its bit rate accordingly to reduce streaming slice's size for shorter transmission time. In a test environment, all streaming slice files can be downloaded in a limited time

TABLE V: HLS performance

Bandwidth(Mbps)	2	1	0.5	0.25
Slice file size(Mbytes)	4.1	4.1	4.1	4.1
Transmit time()	2.05	4.1	8.2	16.4

TABLE VI: AHLS performance

Bandwidth(Mbps)	2	1	0.5	0.25
Slice file size(Mbytes)	5.688	3.535	1.247	0.803
Transmit time()	2.844	3.535	2.493	3.229

for fluent streaming playback and better viewer experiences. Fluent streaming playback at the expense of poorer image quality is acceptable. The AHLS design fits full range of bandwidth variation to relieve users from worrying about their playback environment.

VI. Conclusion

This paper is aimed at adjusting video bit rate dynamically according to existing bandwidth and providing online streaming video at the optimum image quality with limited image encoding parameters. The following conclusions are reached from studies described herein.

1) The test results indicates that the bit rate of video can be adjusted dynamically according to changing bandwidth between the client end and server end to enable users' smooth playback of online streaming in any networking environment including landline, wireless and mobile.

2) The practical web operation interface can work with iPhone, iPad, Android 4.0 mobile phone and Mac computer to provide the same operation interface for fluent playback. It plays online streaming without a client end program. It also plays in a browser without any plug-in like Adobe's Flash.

3) It may extend its application to create mobile theater environment by connecting tablet computer to TV or projector.

Making the most of known existing bandwidth by adjusting bit rate effectively is a critical task. Current formulas can deal with most bandwidth variation scenarios. It may be better to have a mechanism to estimate changes in bandwidth so that self adaptive bit rate for video encoding can be available before the bandwidth is known. This not only frees the system from waiting for bandwidth detection but also helps in getting closer to instant encoding synchronized with video playing. The result is more efficient bit rate adjustment and immune from impacts of fluctuating bandwidth variation environment.

As only HLS playback is supported now, available playback platforms are limited. The program in this paper has taken cross platform versatility into account. It is expected to add

more platform supports in future for fully cross platforms online streaming system.

REFERENCES

[1] D. Wu, Y. T. Hou, W. Zhu, Y.-Q. Zhang, and J. M. Peha, "Streaming video over the internet: approaches and directions," in *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 11, 2001, pp. 282–300.

[2] Report: Video accounts for half of all mobile traffic; android biggest for mo- bile ads. [Online]. Available: http://tinyurl.com/82svkle

[3] T. Lohmar, T. Einarsson, P. Fröjdh, F. Gabin, and M. Kampmann, "Dynamic adaptive http streaming of live content," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*, 2011, pp. 1–8.

[4] C. Y. Hung, "Live broadcast and vod system based on http live streaming," , 2012.

[5] S. K. Hsieh, "Research and design of video adaption algorithm based on http live streaming," , 2012.

[6] Webvtt: The web video text tracks format. [Online]. Available: http://dev.w3.org/html5/webvtt/

[7] Video subtitling and webvtt. [Online]. Available: http://html5doctor.com/video-subtitling-and-webvtt/

[8] Http live streaming examples. [Online]. Available: https://developer.apple.com/resources/http-streaming/examples/

[9] Apple http live streaming. [Online]. Available: https://developer.apple.com/resources/http-streaming/

[10] K. J. Ma and R. Bartos, "Http live streaming bandwidth management using intelligent segment selection," in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, 2011, pp. 1–5.

[11] T. Kupka, P. Halvorsen, and C. Griwodz, "An evaluation of live adaptive http segment streaming request strategies," in *Local Computer Networks (LCN), 2011 IEEE 36th Conference on*, 2011, pp. 604–612.