P2P SVC-encoded Video Streaming Based on Network Coding

李昱憲
國立中正大學資訊工程所
lyh96m@cs.ccu.edu.tw

鄧鴻毅
國立中正大學資訊工程所
thy95p@cs.ccu.edu.tw

黃仁竑
國立中正大學資訊工程所
rhhwang@cs.ccu.edu.tw

***摘要*—隨著網際網路技術的快速發展以及寬頻網路的普及化，視訊串流服務已成為一個殺手級的應用服務。為了解決裝置多樣性問題，Joint Video Team 與 MPEG 共同制定了一個新的影片編碼技術，可調性視訊編碼(Scalable Video Coding, SVC)。使用 SVC 編碼視訊串流，每種裝置皆可根據本身的能力解碼出適合播放的視訊串流。此外，由於點對點傳輸技術相較於傳統的主從式架構具有高擴充性、高彈性以及能避免視訊串流伺服器成為傳輸瓶頸等優勢。近年來許多研究針對點對點技術傳輸視訊串流提出了解決方案。無論如何，大部分的研究並未將 SVC 的特性納入設計考量。因此在本研究中，我們提出了一個 P2P SVC-encoded Video Streaming System。首先，我們針對 SVC 的特性提出了一個新的網路編碼(Network Coding, NC)方法，SVC-NC。SVC-NC 能結合 SVC 與 NC 的優勢，讓視訊串流能夠滿足異質性設備的需求，以及具備高傳輸效率、抗干擾等特性。進一步地，我們提出了三種排程機制來有效地提升視訊串流的傳輸品質。最後，我們透過模擬分析，證明了我們的系統能提供使用者一個高連續播放、低緩衝時間的視訊串流服務，並且能夠有效的減少封包遺失的影響。**

***關鍵詞*—點對點傳輸技術、視訊串流、網路編碼、可調性視訊編碼、封包遺失**

一、簡介

隨著網際網路技術的快速發展以及寬頻網路的普及化，視訊串流服務已成為一個殺手級的應用服務。除了個人電腦外，越來越多種類的裝置能夠收看網際網路上的視訊串流。由於各種裝置的計算與視訊播放能力都不盡相同，促使單一畫質的視訊串流不能滿足各式各樣的裝置。因此有些視訊串流系統，如：YouTube[23]和土豆網[19]，提供了兩種畫質(一般畫質與高畫質)的視訊串流讓使用者自行選擇。但還是無法完全解決裝置多樣性(device diversity)的問題。Joint Video Team 與 MPEG 共同制定了一個新的影片編碼技術，可調性視訊編碼(Scalable Video Coding,SVC)。SVC 是 H.264/AVC 的延伸標準並且能向下相容 H.264/AVC。它能讓視訊串流擁有三個維度(Temporal、Spatial、Quality)的延展性，因此裝置可以根據本身的能力解碼出適合播放的視訊串流。例如手機與 PDA 等螢幕較小且播放能力較弱的裝置就可以只播放基礎畫質的影片。對於伺服端來說，視訊串流也只需要經過一次編碼即可。

在傳統的主從式架構下，面對使用者日漸增多的情形，視訊串流服務提供者為了維持一定的服務品質，必須負擔大量的視訊串流伺服器、網路線路以及機房租金等建置成本。以 YouTube 為例，每個月都必須支付超過一百萬美金的頻寬費用給 ISP 業者[3]。因此點對點(Peer-to-Peer, P2P)傳輸技術成為了一個實用的解決方案。在點對點傳輸技術中，參與節點相互分享本身所有的視訊串流為主要傳輸方式，不再只單靠視訊串流伺服器來提供視訊串

流的傳輸。所以 P2P 傳輸技術具有高擴充性(Scalability)、高彈性(Resilience)以及能避免視訊串流伺服器成為傳輸瓶頸等特性。

無論如何，使用 P2P 傳輸技術傳輸可調性視訊編碼的視訊串流仍然面臨了許多嚴峻的挑戰。像是面對使用者動態地加入與離開的現象(Churn effect)，如何讓每個節點都能先接收到基礎畫質的視訊串流維持視訊播放，以及如何讓每個節點都能根據本身的能力接收到最高畫質的視訊串流。近年來，許多相關研究[8][20][21][22]針對點對點傳輸架構與參與節點的特性提出了解決方案來改善視訊串流服務的品質。其中，我們發現許多學者使用了通道編碼(Channel coding)技術[4][12][13]，如: 網路編碼(Network Coding, NC)[17]、噴泉碼(Fountain Code) [1][9]，來提升視訊串流的傳輸有效性，以及減緩 Churn 所造成的不穩定性。無論如何，大部份的研究並沒有針對 SVC 的特性提出解決方案。

因此，在本論文中，我們植基於 NC 提出了一個 P2P SVC-encoded video streaming system。首先，我們針對 SVC 的特性提出了一個新的 NC 方法，SVC-NC。SVC-NC 能結合 SVC 與 NC 的優勢，讓視訊串流能夠滿足異質性設備的需求，以及具備高傳輸效率、抗干擾等特性。進一步地，我們運用 SVC-NC 的特性提出了三種排程方式，分別為 Startup request scheduling、Priority request scheduling、Priority response scheduling，來有效地提升視訊串流的傳輸效率。最後，我們透過模擬分析來驗證我們的系統可以達到 High quality、Low startup latency 以及 Low packet redundancy。

本論文共分為五個章節，第一章闡述研究背景、研究動機、與研究目標。第二章介紹 P2P 視訊串流的相關研究。第三章介紹我們所設計的系統。第四章介紹我們使用的模擬器和模擬設定，並且分析實驗結果。最後一章提出我們的結論以及未來研究方向。

二、相關文獻

P2P 視訊串流研究以架構來區分可分為二大類：Tree-based、Mesh-based 以及 Hybrid。Tree-based 的方法[8][20][22]傳輸資料的方式都以 Push-based mechanism 的為主，能達到低傳輸延遲的視訊串流品質。但是 Tree-based 的方法在 Churn effect 嚴重時，需要耗費大量成本去維護樹狀結構的完整性。修補樹狀結構的同時，視訊串流的傳輸效能下降的十分快速，甚至是中斷連線。而 Mesh-based 的方法，如: CoolStreaming[21]則是藉由節點定期交換的 buffer map，向其他節點要求(Pull-base mechanism)所缺乏的視訊串流。Mesh-based 的方法相較於 Tree-based 的方法有較佳的彈性與強韌性[14]。目前已上線的 P2P 視訊串流系統，如： PPLive[15]、PPStream[16] 以及 SopCast[18]都是 Mesh-based。因此，現今的研究趨勢是以 Mesh-based 為主流。另外，也有研究[5]利用節點的特性提出 Hybrid 架構，它將穩定的節點建構成一個 Tree 的架構，其餘的節點則維持 Mesh 架構，使得 Tree-based 與 Tesh-based 的優勢能並存。

除此之外，有研究[2][7][12][13]使用 NC 來提升視訊串流的傳輸效率。在[12] 中，作者植基於 Randomized network coding，在 Mesh-based 的架構上提出了 Push-based 的排程機制來提升傳輸效率。而在[13]中，作者在 Mesh-based 的架構上提出了 Random push 的排程機制降低 Buffering delay、Server 的頻寬負擔、以及 Churn effect。無論如何，上述的研究都沒有將 SVC 納入設計考量。在[10]中，基於 SVC 的特性，作者提出了 Peer selection、Data flow model、Buffer management 和 Scheduling mechanism 等機制，讓用戶端能流暢地播放 SVC 視訊串流。

三、系統設計

節點的系統架構如圖 1 所示，主要有三大模組: (1) Membership Manager 負責管理其他節點的資訊；(2)Scheduler 負責排程 SVC 視訊串流的傳輸；(3)Network Coding Encoder/Decoder 負責編碼傳輸前的 SVC 視訊串流以及解碼接收到的 SVC 視訊串流。當一個節點加入系統時，會先跟 Boot-strap node 要求一份隨機產生的節點清單。根據節點清單，新加入的節點與其他節點開始定期性地交換 Buffer map，接收到的 Buffer map 會儲存在 Cache 裡。藉由儲存在 Cache 裡的 Buffer map，新加入節點開始以 Pull-based 的方式向其他節點要求視訊串流，接收到視訊串流會先經過 NC 解碼後儲存在 Buffer 裡。最後播放器會將 Buffer 裡的視訊串流解碼後播放給使用者觀看。

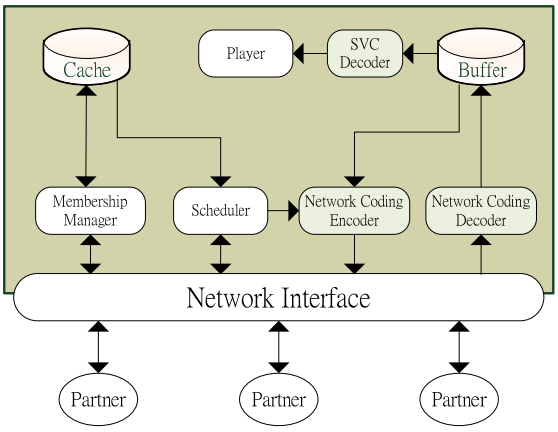

圖 1 Node 的系統架構圖

(一) SVC-NC

為了讓 SVC 視訊串流除了能滿足異質性設備的需求，還可以具備高傳輸效率、抗干擾等特性，我們提出了一個新的 NC 方法，SVC-NC。如圖 2 所示，我們根據 SVC 視訊串流中解碼的時間關係，將每個 Group Of Picture(GOP)區分為數個 Layer。每個 Layer 的 Frames(包含 Spatial 與 Quality 維度)分別使用 Randomized Network Coding 編碼成相對應的 Code word。因此在整個系統中的 Code word 會依據 SVC 的 Layer 分為數種，如圖 2 中一個 GOP 有四個 Layer，那整個系統中就會有四種的 Code word。

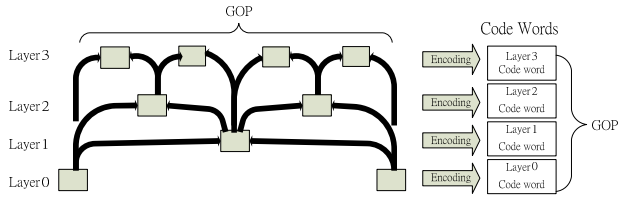

圖 2 SVC-NC 編碼示意圖

根據 SVC-NC 的設計，節點上 Buffer 的設計如圖 3 所示。Current Play Location(CPL)為目前播放的時間點，在 CPL 位置前我們保留了一段已觀看過的視訊資料，提供給其他節點下載。此外，我們使用 Sliding window 來做為排程的範圍限制，每下載完一個 GOP 或是當 CPL 到達 Sliding window 的起始點，Sliding window 就會向後滑一個 GOP。如圖 4 所示，一個 GOP 對應到 *n* 個 Segments，一個 Segment 對應到一個 SVC-NC 的 Layer，一個 Block 對應到一個 SVC-NC 的 Layer 的 Code word。在播放時，一個 GOP 中，Base layer (Layer 0) 的 Code word 必需要能被成功的解碼還原才能順利的播放，而更多 Layer 的 Code word 被解回來時，就可以結合 Base layer 的資料產生品質更好的視訊，反之若 Base layer 的資料無法被解回來時，就算其他非 Base layer 的資料能夠還原也是沒辦法獨自播放。

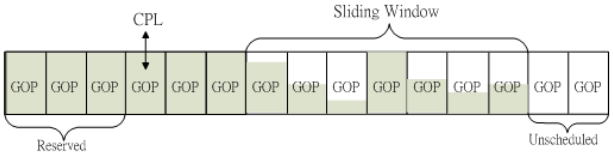

圖 3 Node 的 Buffer 示意圖

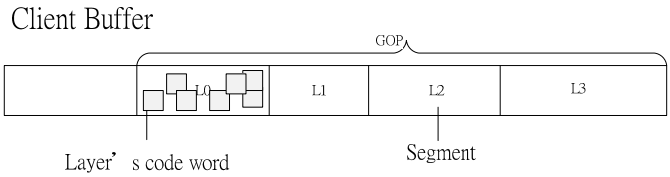

圖 4 GOP 示意圖

如圖 5 所示，在 Buffer map 的表示是以一個 GOP 為基準。一個 GOP 裡的每個 Layer 會使用 1 bit 來表示。以 4 個 layer 舉例來說，一個 GOP 的 Buffer map 為 4 個 bit。根據[11]的研究，它們指出當 Code word 的數量小於 66%時，傳輸效能是不好的。所以本論文的 Buffer map 表示，當一個 layer 的 code word 數量小於 66%時為 0，還不足以提供其他節點這個 layer 的資料；當大於 66%時為 1，表示可以開始接受別人的傳輸要求。

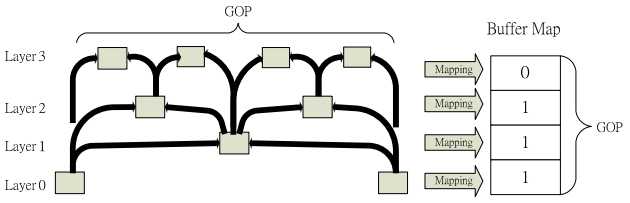

圖 5 Buffer Map 示意圖

(二) Scheduling Mechanisms

針對 SVC-NC 的特性，我們提出了三種排程的方式來提升其傳輸效能。其中 Startup request scheduling 與 Priority request scheduling 是要求者的角度做設計，而 Priority response scheduling 則是以回應者的角度來設計。在兼顧要求者與回應者的排程設計下，我們預期 SNC-NC 的優勢將能全面性的發揮。

■ Startup Request Scheduling

Initial buffer delay 是使用者選擇頻道後至使用者收看到視訊的延遲時間。較長的 Initial buffer delay 會降低使用者的滿意度，因此為了減少 Initial buffer delay，我們透過 SVC 只需要擁有 Base layer 就可以播放的特性，設計了以下的排程機制。如圖 6 所示，在 Initial buffering 階段，每個節點只下載 GOP 裡的 Base layer。每當一個 GOP 的 Base layer 已完成下載，Buffer 裡的 Sliding window 會自動往下移動一個 GOP，直到 *S* 個 GOP 都完成下載。

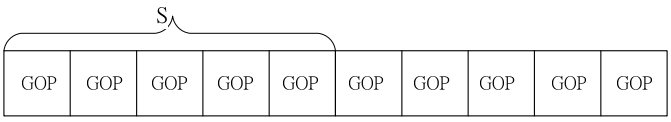

圖 6 Initial Buffering 示意圖

■ Priority Request Scheduling

離開 Initial buffering 階段，每個節點開始播放視訊串流。為了讓節點能維持播放的連續性以及能根據自己的能力下載到最高品質的視訊串流。透過 SVC-NC 裡每個 Layer 的 Priority 不同的特性，我們設計以下的排程機制。如圖 7 所示，針對 Sliding window 裡的每一個 GOP，節點會循序地先要求 Priority 較高的 layer，如 :GOP_1 的 $Segment_1$➔GOP_2 的 $Segment_1$➔GOP_3 的 $Segment_1$➔GOP_4 的 $Segment_1$，再要求 GOP_1 的 $Segment_2$➔GOP_2 的 $Segment_2$➔GOP_3 的 $Segment_2$➔GOP_4 的 $Segment_2$，依此類推。每種 Layer 的 Segment 會隨機選出 m 個節點並向這些節點要求下載。除此之外，節點會根據 Packet loss rate 來計算一次要跟對方請求多少 Code words，以提升傳輸效率。計算方法如(1)所示，i 為第幾個 Layer，N_i 為節點所需要的 Code word 數量、N_{ip} 為我們要求送出 Code word 數量，才能順利接收到大於 N_i 的數量、α 為二個節點之間的封包遺失率、ε 為預期達成目標的機率有多大。舉例來說，原本要向這二個節點分別要求傳 3 個 Code word，而 packet loss rate 為 50%，想要達到 99%的機率能成功，節點可以計算需要傳送出 6 個 Code word。每個節點根據本身的能力下載完一個 GOP 後， Sliding window 就會自動地移動到下一個 GOP。

$$Layer(i): \sum_{k=N_i}^{N_{ip}} C_N^{N_{ip}} \alpha^{N_{ip}-k} (1-\alpha)^k \geq \varepsilon_i \quad (1)$$

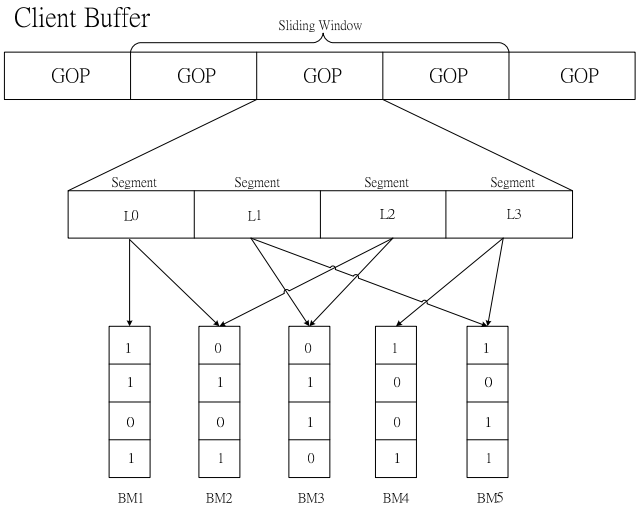

圖 7 Priority Request Scheduling 示意圖

■ Priority Response Scheduling

除了以要求者的角度而設計的兩種排程之外，我們進一步地以回應者的角度設計了以下的排程機制，讓重要 Layer 的 Segment 要求能夠優先被回應。首先，傳輸的請求分成二種：一種為 Startup Request(SR)、一種為 Normal Request(NR)。SR 為 Startup Request Scheduling 的 Request，而 NR 為 Priority Request Scheduling 的 Request。NR 又細分為不同 layer 的 Request，如：GOP_2 的 $Segment_1$(L0)、GOP_3 的 $Segment_2$(L1)。接著傳輸的請求依照優先權的特性分為 L 個 Queue(L 為 SVC-NC 的 layer 數)，以重要性高的 Queue 先回應，Queue 的編號越大比較裡面的請求重要性越小。以一個節點的上傳頻寬為 300kbps，有三個 Queue 為例，$Queue_1$ 為 NR(L0)的請求、$Queue_2$ 為 NR(L1)的請求、$Queue_3$ 為 NR(L2)的請求，SR 的請求將隨機放在任一 Queue 中，排程機制為(1)一次處理一個 Queue 中的所有請求，每處理一個請求時會消耗相對的頻寬；(2)當一個 Queue 中的請求已處理完，就換下一個 Queue 直到頻寬用完為止或是所以請求處理完畢。以圖 8 所示，假設 NR(L0)的請求需要額度 10 頻寬、NR(L1)的請求需要額度 30 頻寬、NR(L2)的請求需要額度 50 頻寬、SR 的請求需要額度 10(base layer 的 size 最小)。那麼第一次排程結果為 $Queue_1$ 的 11 個請求，加上 $Queue_2$ 的 6 個請求。

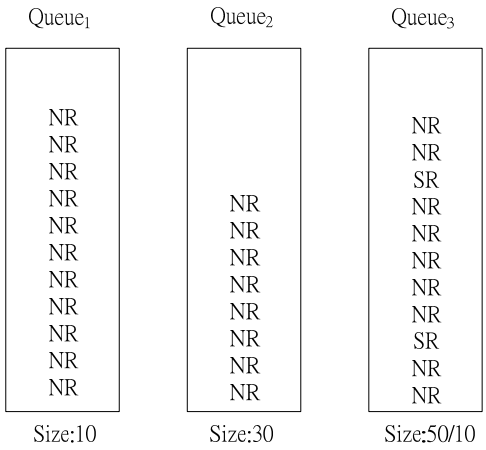

圖 8 Priority Response Scheduling 示意圖

這個機制可以讓 Priority 高的請求有較高優先權被處理。對應於可調性視訊編碼特性，Base layer 的資料是最重要的，而且可以直接播放，所以需要有較高優先權的處理。而 SR 的 Request 是希望不在忙碌中或頻寬足夠的節點才回覆這個請求，所以使用隨機的方式放置在 Queue 中，這樣的好處是有一定的機率讓目前是不忙碌的節點才能即時的處理這個請求，又能防止 SR 的 Request 不會在所有節點都在忙碌的情況下發生 Starvation，使得新節點在找其他節點時能夠做到負載平衡。

四、模擬分析

(一) 模擬環境

我們使用 Event-driven packet-level 的模擬器[6]，底層的網路拓樸使用 GT-ITM 來動態產生，有 2500 個 Node，以及定義好了每個 Node 之間的延遲。此外，任二點之間的 Packet loss rate 為隨機產生小於 10%的數值。在可調式視訊串流的設定，如圖 9 所示，這個可調性視訊編碼視訊分為 10 個 Layer。如表 1 所示，參與節點有 4 類(不包含 Source node)，對應的比例為 1%、15%、39%以及 45%。使用者的 Buffer 存放二分鐘的視訊資料，Sliding window 的大

小為 10 秒，若 Buffer 已滿，看過的視訊將保留 20 秒。

Layer	Resolution	Framerate	Bitrate	DTQ	Max NALUnit Size(bit)
0	352x288	1.875	19.1	(0,0,0)	1049
1	352x288	3.75	30.4	(0,1,0)	961
2	352x288	7.5	45.8	(0,2,0)	656
3	352x288	15	64.9	(0,3,0)	429
4	352x288	30	87.1	(0,4,0)	299
5	352x288	1.875	387.3	(0,0,1)	22850
6	352x288	3.75	489.7	(0,1,1)	5944
7	352x288	7.5	615.6	(0,2,1)	3778
8	352x288	15	762.5	(0,3,1)	2164
9	352x288	30	909.6	(0,4,1)	1578

圖 9 視訊的各項參數

表 1 節點種類分配表

	Inbound BW	Outbound BW	FRACTIONS
Source Node		600 Kbps	
Type 1	10 Mbps	5 Mbps	0.01
Type 2	3 Mbps	1 Mbps	0.15
Type 3	1.5 Mbps	384 Kbps	0.39
Type 4	768 kbps	128 Kbps	0.45

(二) Continue Index

首先，我們透過 Continue Index 來驗證影片播放的流暢性，Continue Index 值介於 0~1 之間，越接近 1 表示使用者在觀看上比較不會有停頓的情況，計算的方式採用在一段時間內，應該要播放的 GOP 數量除以可以播放的 GOP 數量。從圖 10 來看，我們的方法，因為較重要的 Layer 都有被優先處理，並且 Base layer 的數量比較少，所以可以很迅速的被排程機制處理完，讓使用者能夠順利的播放。而 Pure SVC 的情況約有 60%的使用者他的 Continue Index 都在 0.9 以下，從數據中顯示，大都為 Type3 和 Type4 的使用者，但以 Type3 和 Type4 使用者的頻寬設定來看，應該是足夠下載這個視訊，會造成播放不連續的原因主要是因為在判定自己所需要的片段要和哪一個使用者索取時，會因為 Buffer Map 的交換延遲，使得資料不夠急時，或是傳輸過程中因為封包遺失導致要重新排程等待，所以從圖中我們可以發現，採用 NC 真的能有效的改善以上的情況。而圖 11 表示整個模擬時間內，所有使用者的平均 Continue Index，在第一分鐘時，二組實驗數據都有不錯的表示，這是因為在實驗設定中，設定為前 10 秒的視訊要全部抓完才能開始播放，或是當使用者加入後 10 秒後也會直接開始播放，避免使用者 buffer 過久導致跟不上整體視訊的播放。

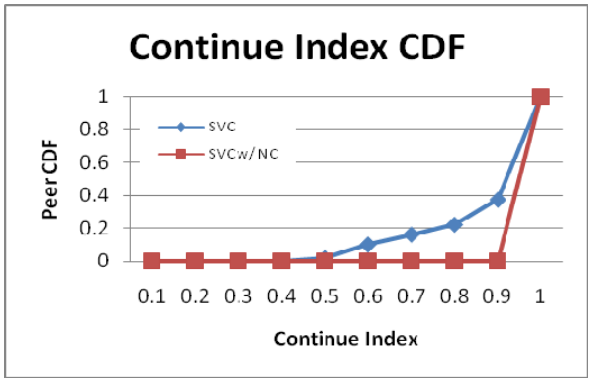

圖 10 Continue Index CDF 圖

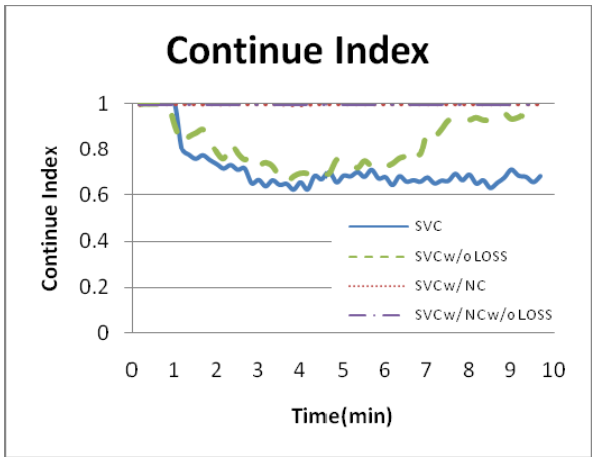

圖 11 Continue Index 比較圖

(三) Start Play Time and Buffering Time

接著，我們觀查使用者等待直到可以觀看到第一個畫面的時間，我們將它稱之為等待開始播放時間(Start Play Time, SPT)。越短的 SPT，表示使用者可以在進入系統後短時間內看到視訊串流。另一個相對應的效能參數為緩衝等待時間(Buffering Time)，它是由使用者接收到第一個資料開始，不考慮前面的排程時間，直到可以觀看到第一個畫面的時間。從圖 12 與圖 13 中可以發現，加入了我們設計的方

法後，Start Play Time 平均所有的使用者減少了大約 1~2 秒的時間；Buffering Time 平均所有使用者減少了 3~4 秒的時間，這二個數據的差別主要是 Start Play Time 還有加入使用者剛加入後的排程和交換 Buffer map 的時間，這部份的等待時間比較隨機，若不考慮排程的時間，單從 Buffering Time 這邊可以看的出來，我們所提的方法能夠讓使用者進入系統後很快地收到看視訊串流。

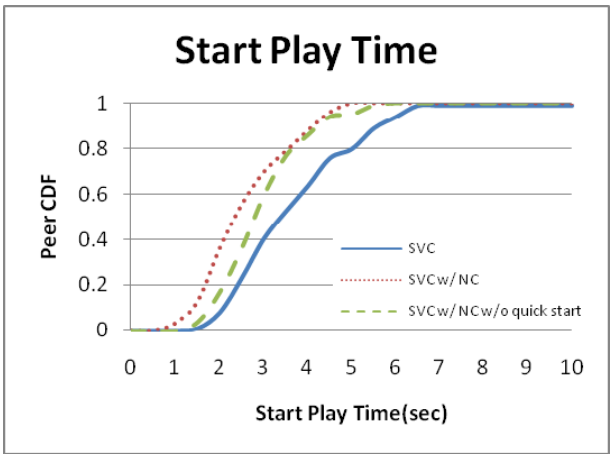

圖 12 Start Play Time 比較圖

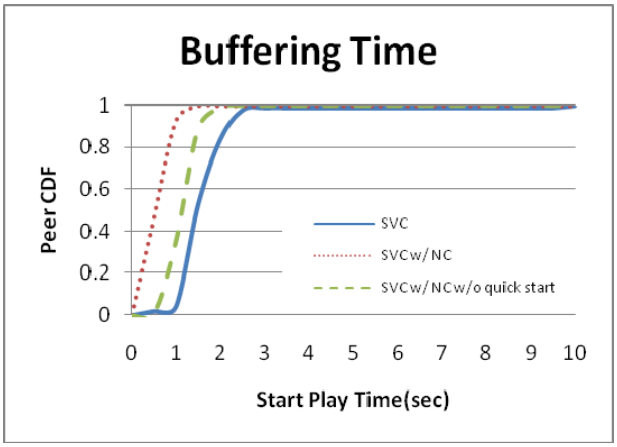

圖 13 Buffering Time 比較圖

(四) Quality

從上面的數據我們證實了我們所提的方法的有效和正確性，現在我們要看當我們的方法是否能提升視訊品質。視訊品質是使用相對應的 Layer 值來表示，Layer 數值越高，表示視訊品質越好。從圖 14 中來看，我們可以發現在第一分鐘和第二分鐘二組數據有交叉的地方，一開始是我們所提的方式的視訊品質較差，而 Pure SVC 的品質較好，這是因為二者在一開始都有做 Buffering 的動作，但是我們所提的方式在一開始的 buffering 時間內，所要求的都只是 base layer，為了能夠更快速的有畫面的輸出，所以我們選擇降低了前面一段時間的視訊品質，才會使得前面一段時間內的數據較差，但在一段時間過後，因為網路編碼技術在傳輸上的幫助，所以整體的視訊品質會開始上升。Pure SVC 的數據一開始是表現良好的，這也是因為一開始的 Buffering 動作，但開始播一段時間過後則是因為本身缺乏排程，所以視訊的接收和解碼會不如我們所提的方式，整體的視訊品質就會開始往下降，到後來則慢慢趨於平穩。

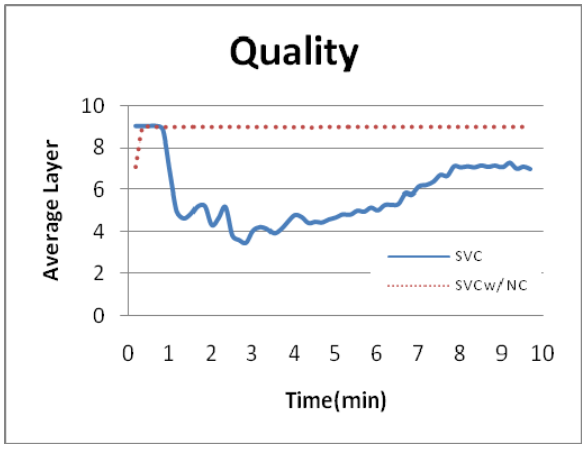

圖 14 Quality 比較圖

(四) Packet Redundancy

最後，由於我們的方法使用了 Packet loss rate 來計算 code word 的數量，我們從 Packet Redundancy 來驗證我們的方法是否會產生明顯的 Overhead。如圖 15 所示，對於整體的 Packet redundancy overhead 只有在 Packet loss rate 大於 25%時產生比較明顯的差異。當 Packet loss rate 小於 20%時，我們的方法將不會對系統帶來額外的 Packet redundancy。

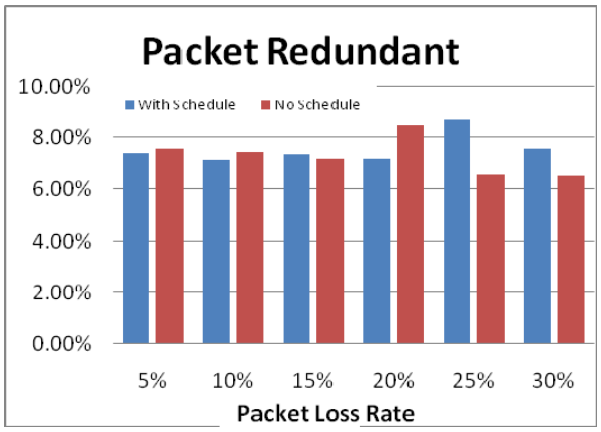

圖 15 Packet Redundancy 比較圖

五、結論與未來展望

在本研究中，我們植基於 NC 提出了一個 P2P SVC-encoded Video Streaming System。首先，我們提出了一個新的 NC 編碼方法，SVC-NC。SVC-NC 能結合 SVC 的特性與 NC 的優勢，讓視訊串流能夠滿足異質性設備的需求，並具備高傳輸效率、抗干擾等特性。除此之外，我們提出了三種排程機制，分別為 Startup request scheduling、Priority request scheduling、Priority response scheduling。Startup request scheduling 能讓用戶端在很短的時間內看到視訊串流。而 Priority request scheduling 與 Priority response scheduling 讓重要性高的 Segment 能優先完成下載，提升視訊串流服務的品質。在模擬分析的部份，模擬結果顯示我們所提出的系統能讓所有的節點在 5 秒內就能播放視訊串流。此外，除了 Initial buffering 階段，所有的節點都能接收到最高畫質的視訊串流，而且不會產生過多的 Redundant packets。未來，我們將實做我們所提出的系統，透過 PlanetLab 的環境來進一步地實證系統效能。

六、誌謝

感謝國科會計畫 NSC 97-2221-E-194-011-MY3 及 NSC 97-2221-E-194-012-MY3 提供經費支持本研究的進行。

七、參考文獻

[1] A. Shokrollahi, “Raptor Codes,” Proc. IEEE Int’l Symp. Information Theory, June 2004.

[2] C. Gkantsidis and P. Rodriguez, “Network Coding for Large Scale Content Distribution,” in Proc. of IEEE INFOCOM, March 2005.

[3] C. Huang, J. Li, and K. W. Ross, “Can Internet Video-on-Demand be Profitable,” In Proc. of ACM SIGCOMM, August 2007.

[4] C. Wu and B. Li, “*rStream*: Resilient and Optimal Peer-to-Peer Streaming with Rateless Codes,” IEEE Transactions on Parallel and Distributed Systems, vol.19, no.1, pp. 77-92, Jan. 2008.

[5] F. Wang, J. Liu, Y. Xiong, “Stable Peers: Existence, Improtance, and Application in Peer-to-Peer Live Video Streaming,” in Proc. of IEEE INFOCOM, pp. 2038-2046, 2008.

[6] P2PStrmSim. [Online]. Available. http://media.cs.tsinghua.edu.cn/~zhangm/

[7] K. Nguyen, T. Nguyen, and S. Cheung, “Peer-to-Peer Streaming with hierarchical network coding,” IEEE ICME, July 2007.

[8] M. Castro, P. Druschel, A. M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, “SplitStream: High-Bandwidth Multicast in Cooperative Environments,” in Proc. of the 19th ACM Symposium on Operating Systems Principles (SOSP’03), Oct. 2003.

[9] M. Luby, “LT Codes,” Proc. 43rd Symp. Fundations of Computer Science, Nov. 2002.

[10] M. Mushtaq and T. Ahmed, “Smooth Video Delivery for SVC based Media Streaming over P2P Networks,” 5th IEEE Consumer Communications and Networking Conference, pp. 447–451, 2008.

[11] M. Wang and B. Li, “How Practical is Network Coding?,” in Proc. of the 14th IEEE International Workshop on Quality of Service (IWQoS 2006), June 2006.

[12] M. Wang and B. Li, “Network Coding in Live Peer-to-Peer Streaming,” IEEE Transactions on Multimedia, vol. 9, pp. 1554–1567, Dec. 2007.

[13] M. Wang and B. Li, "R^2: Random Push with Random Network Coding in Live Peer-to-Peer Streaming," IEEE Journal on Selected Areas in Communications, vol. 25, pp. 1655–1666, Dec. 2007

[14] N. Magharei, R. Rejaie, and Y. Guo, "Mesh or Multiple-Tree: A Comparative Study of P2P Live Streaming Services," in Proc. of IEEE INFOCOM, 2007.

[15] PPLive. [Online]. Available: http://www.pplive.com/

[16] PPStream. [Online]. Available: http://www.ppstream.com/

[17] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, "Network Information Flow," IEEE Transactions on Information Theory, vol. 46, no. 4, pp. 1204–1216, July 2000.

[18] Sopcast. [Online]. Available: http://www.sopcast.org/

[19] Tudou. [Online]. Available: http://www.tudou.com/

[20] V. Venkataraman, P. Francis, and J. Calandrino, "Chunkyspread: Multi-tree Unstructured Peer-to-Peer Multicast," in Proc. of the 5th International Workshop on Peer-to-Peer Systems, Feb. 2006.

[21] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "DONet/CoolStreaming: A data-driven overlay network for live media streaming," in Proc. of IEEE INFOCOM, 2005.

[22] Y. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," in Proc. ACM SIGMETRICS, pp.1-12, 2000.

[23] Youtube. [Online]. Available: http://tw.youtube.com/