在 CDN-P2P 網路上採用跨串流描述修復提升影音服務的堅韌性*

林朝興	李明憲	王鼎超
資訊工程系所	資訊管理系所	資訊管理系所
國立台南大學	南台科技大學	南台科技大學
副教授	碩士班研究生	助理教授
mikelin@mail.nutn.edu.tw	m9690237@webmail.stut.edu.tw	zh9@mail.stut.edu.tw

摘要

隨著網路的頻寬快速提高，更多使用者直接在隨選視訊伺服器中點選喜歡的影片來進行觀看，如何有效率的提供多媒體影音串流服務，變成一項非常值得關注的議題。在本研究中在 Loopback-MDC 架構上，提出 S&P 錯誤回復機制，來解決當 client 中途離開時，造成代理伺服器負載提高的議題。在本論文中，我們管理 client 中的暫存空間以及採用跨串流描述修復機制提出 S&P 修復法，當執行 S&P 修復法時，考量 Loopback-MDC 中的堅韌度和可用度，來提升代理伺服器的延展性，降低使用者的中途離開而影響其他使用者的觀看，並提供使用者更好的多媒體影音串流服務。模擬實驗結果顯示 S&P 錯誤回復機制在 Loopback-MDC 架構上可以更有效率降低使用者中途離開所造成 CDN-P2P 隨選視訊代理伺服器的上傳頻寬。

關鍵詞：*錯誤回復, 同儕式網路, 代理伺服器, 緩衝區管理**

Abstract

With the rapidly increasing bandwidth at the access link of end users, more and more users nowadays tend to select and enjoy their desired videos directly from VoD servers. How to efficiently provide on-demand media streaming services has emerged as an important issue deserving serious attention. In this paper, based on the Loopback-MDC architecture, a failure recovery mechanism, S&P, is proposed to reduce the additional workload imposed on the proxy server due to clients' early departures. The proposed S&P failure recovery mechanism includes client buffer management and inter-description recovery. Both the robustness and availability of video streams are taken into account in the design of the proposed S&P failure recovery mechanism so as to boost the scalability of proxy servers, lessen the impact of clients' early departure on others and provide users with better QoS. The simulation results show that the S&P recovery mechanism can efficiently deal with clients' early departures and reduce the bandwidth consumption required for the proxy server to recover clients abandoned by departing ones.

Keywords: Failure Recovery, multiple description code, peer-to-peer network, proxy server, buffer management

1. 前言

* 本研究承蒙國科會部份贊助，計劃編號：NSC 97-2221-024-014-MY3

VoD server 主要在提高系統的延展性(scalability)，當有大量的 client 觀看請求到達(flash crowd) 或大量 client 離開情況時，server 希望有足夠的頻寬和儲存空間來解決這些突如其來的情況。傳統的 Client-Server 架構，採用的是集中式管理方法，優勢在於確保品質，但當有大量請求，系統無法負荷，可能導致使用者觀看的滿意度會降低。後來有許多學者針對所發生的情況提出了一些改善的作法，例如 Batching[2]、Patching[8]等技術，但上述所提到的數種作法皆建立在 IP multicast 上，關係到實體層面。依據目前已存在的網路架構而言，並非是一種良好的改善方式。因此學者後續進一步提出 CDNs(Content Distribution Networks)架構概念，即在現有的網路基礎架構中，以區域為單位設立多台 proxy server，並將影片內容放置於中，藉由這些 server 來分擔單一 server 的負載，避免無法負荷。但要增加延展性，須大量提高成本與資源。而後來的 Peer-to-Peer 架構[14][5]可以解決大量請求的問題，P2P 運用 client 為服務端亦為接受端的特性，使 client 的暫存空間中的影片內容，可以與其它 client 進行影片傳送，支援 server 固定的頻寬，進一步降低 server 的負載。當大量的 client 請求出現，可以提供大量的頻寬來支援現有的 server 頻寬，如此一來將提高 VoD server 的延展性，但是 P2P 的劣勢在於 client 的加入以及離開，並非 server 可以控管。因應上述的問題，有學者提出 Loopback 機制[6]應用在 CDN-P2P 混合架構上[3][4][12]，利用了現有的架構，並排除其缺點，和具有高擴充性。但在 Loopback 機制上，只提供了單一串流來服務 client，並且 client 之間存在網路異質性的問題，造成 server 要把影片設定為最低的串流品質，才能讓大部份的 client 互相順利傳遞串流。基於上述的原因，有學者後續進一步提出 Loopback-MDC 機制[1]，在 Loopback 中利用 MDC 編碼技術將影片編碼成多條 description 串流，解決 Loopback 中的網路異質性問題，且在 Loopback-MDC 的研究成果上，經由實驗結果證實可以有效地提高 Loopback 的延展性。

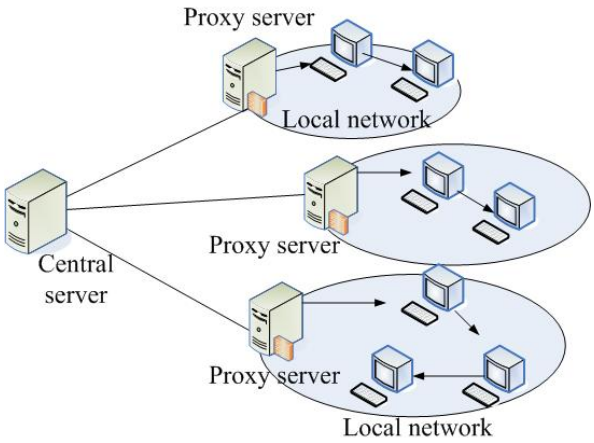


圖 1 CDN-P2P 混合架構

在CDN-P2P中有central server (中央伺服器)、及多台proxy server (代理伺服器)，如圖 1。central server存放所有的影片來源，並提供proxy server影片內容以及支援proxy server來給予client影片內容；proxy server建構在各區域中，並負責cache比較熱門的影片，提供區域下的client進行影片傳送的服務。若client所請求的影片，proxy server已持有，則可直接由它來服務。proxy server 持有熱門的影片，可以提供大多數的client所需的服務，進而降低central server的負載程度。透過同區域和P2P的特性，使得client可以互相支援，讓影片內容可以更快速的進行分享和穩定傳送內容。當有多個client觀看同一部影片時，由最早觀看的client來跟proxy server取得影片內容的片頭，再由client之間進行服務，不需要再經由proxy server或central server來服務，進而可以提高服務的數量。

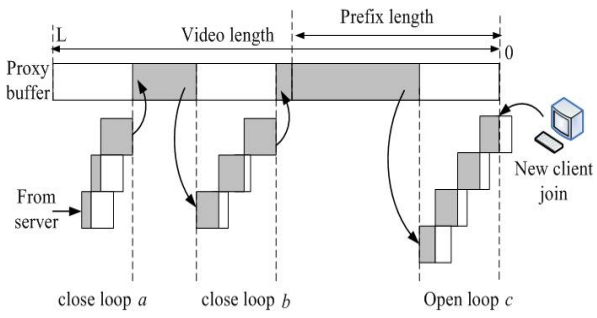


圖 2 Loopback 示意圖

在Loopback 機制中，迴圈運作中有二種情況。第一種稱為開放式迴圈，迴圈呈現可讓新的client 加入的開啟狀態，如圖2的迴圈 c。迴圈 c 中的最後一個client 的buffer 尚未被填滿，影片片頭仍存於client buffer 中，因此可以為下一個新進入的 client 服務；第二種稱為封閉式迴圈，迴圈 呈現無法讓新的client 加入的關閉狀態，如圖 2中的迴圈 a。迴圈 a 中的最後一個client 的buffer 已被填滿，影片片頭已被傳回proxy buffer 中。因此，無法為下一個新進入的 client 服務，有新的client 加入則須開啟另一個新的迴圈。Loopback機制串連請求相同影片的client，如同chaining的架構[9]，依序進行串流傳遞。有別一般的樹狀結構，它採用的是依序串連起來的傳輸方式，使其複雜度降低。但Loopback 機制只提供單一版本的串流資料，使得client 間實際運作多媒體傳輸受到限制。而Loopback-MDC機制下運用 MDC 編碼技術使得一部影片變成多個description串流內容，依據每個client所要求的品質，而接收所需的description數量。因此在CDN-P2P混合架構上使用Loopback-MDC機制，可以讓VoD 在提供服務時，有一個穩定及快速傳送的傳遞架構，並強化了VoD最需要的延展性 (scalability)。而在Loopback-MDC機制中的錯誤回復機制僅利用內部迴圈的client之間，是否有無重複的影格資源來進行修復，如果大量client離開，將導致片頭不能長時間的在client間流傳，來服務新進client，導致central server和proxy server負載上升，降低可以服務的數量。上述情況嚴重降低此架構的應用價值。而本研究嘗試為上述問題尋求解決之道，所以在Loopback-MDC架構下，提出新的buffer管理及S&P錯誤回復機制，讓client在離開系統的時候不會造成central server和proxy server的負載上升，並考量其優先修補順序，使得Loopback-MDC架構將可以更有延展性及堅韌度。

論文章節安排如下，在第二節，探討相關研究的文獻與 Buffer 管理；第三節探討如何執行跨串流描述修復機制；第四節探討連續離開的 client 的情況；第五節描述研究的實驗結果與分析；第六節為最後的結論與未來發展。

2. 文獻探討

2.1 Loopback-MDC

Loopback-MDC 採用 CDN-P2P 混合架構，假設進來觀看影片的 client 都擁有暫存影片內容的能力和動態暫存影片內容於 buffer 中，因此 client 能利用 buffer 中的暫存內容進行串流傳輸給其它 client，但 client 在進行傳送時，可能發生失敗或是中途離開 server。

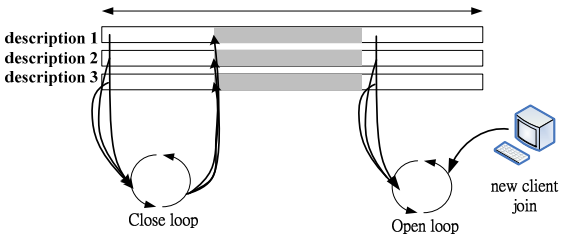


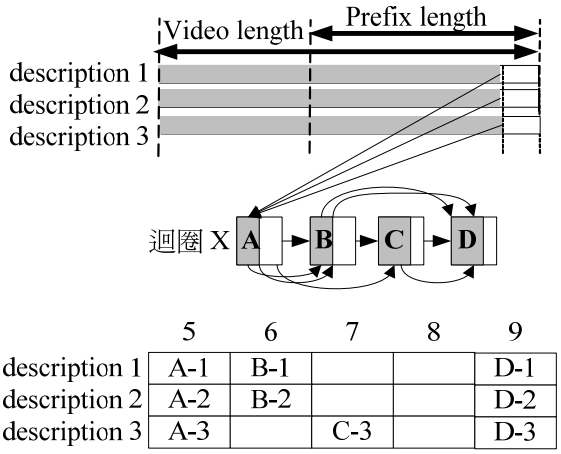
圖 3 Loopback-MDC 示意圖

每個 proxy server 建立在區域網路上，並且暫存較熱門的影片的前半段影片內容，來服務同區網中的 client。在相同區域的 client 觀看一樣的影片時，會依進入的時間，循序產生數個迴圈，當 client 送出請求時，proxy server 提供串流服務，但當 proxy server 無法滿足 client 的請求時，或是 client 已經觀看到影片

片尾時，則由 central server 來提供所需的影片內容。在 proxy server 中的影片內容透過 MDC 編碼機制[7][10][15][11][13]，產生多條的 description，如圖 3。client 可依其請求來接收 description 數量，即 client 接受的 description 數量越多，表示其觀看品質越高；接受到全部的 description 數量，表示此 client 所觀看的影片為最佳品值(full quality)。當第一個進來的 client 根據自身所須的請求從 proxy server 取得數條 description 的影片片頭，來進行觀看。在進行一段時候後，將可服務後來加入迴圈中的 client 當影片片頭一直保持在 client 之內，則表示可以不斷的服務新加入進來的 client，如圖 3 中的開放式 迴圈。因每個 client 所要求的 description 數量不同，並且為了確保影片片頭能夠長時間暫存於 client 的儲存空間中來服務更多的新進 client，應讓迴圈盡可能保持在開放式迴圈的狀態來提供服務，所以在 Loopback-MDC 機制中提出開放式迴圈優先`(OLF) Description 分配[1]。

OLF description 分配法，主要在進行 description 分配時，OLF 先找出為開放式迴圈 的 description，再以這些 description 進行分配讓這些迴圈維持開啟的狀態。因為新開啟一個迴圈，需花費一條 server 的上傳頻寬，而新進 client 所需的 description 數量高於現有開放式迴圈，則由 server 開啟新的開放式迴圈來滿足新進 client 的請求。例如：一部影片共有五條的 description，二條為開放式迴圈，三條為封閉式迴圈，當一個新加入的 peer，要求三條 descriptions 的觀看品質，此時系統會先優先分配二條開放式迴圈的 description 給新加入的 peer ，另在迴圈為關閉狀態的 description，開啟一條新迴圈，讓新加入的 peer 加入，以達到欲觀看的品質。

當開放式迴圈 最後一個 client 的儲存空間填滿，而沒有出現相同請求的 client 而變成封閉式 迴圈，最後一個 client 會把最後的影片片段轉送到 proxy server，當 proxy server 的 buffer 夠大，將使得後續觀看到影片尾的時候，不用跟 central server 請求，而可以直接由 proxy server 來進行服務。



	5	6	7	8	9
description 1	A-1	B-1			D-1
description 2	A-2	B-2			D-2
description 3	A-3		C-3		D-3

圖 4 Loopback-MDC 之 OLF 分配

如圖 4 所示，假設 proxy buffer 暫存某一部影片中的部份內容，並利用 MDC 編碼技術將影片變成三條 description,，每個 client 的暫存空間為三個時間單位。圖中迴圈 X 目前狀態為開放式 迴圈，所以當新的節點在時間單位 9 進來並請求三條 description 時，可以加入迴圈 X。在迴圈 X 中有 client A 請求三條 description、client B 請求二條 description、client C 請求一條 description，經由 OLF 分配，client A 擁有 description 1(A-1)、description 2(A-2)、description 3(A-3)；client B 擁有 description 1(B-1) 、 description 2(B-2) ； client C 擁有 description 3(C-3)，所以當 client D 在時間單位 9 進來時，仍然可以取得它所須三條 description ， 不須由 server 提供新的 description。

2.2 Client Buffer 管理

在 Loopback-MDC 下，client 的儲存空間每從 proxy server 取得一個時間單位內容，就馬上觀看其內容。在發生迴圈斷裂時，進行修

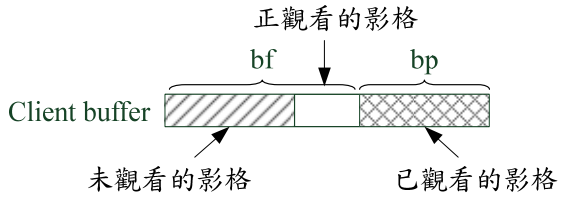
復的期間，因無暫存的影片內容可供使用者觀賞，降低了使用者的滿意度。另外跨串流描述修復機制在進行修復的期間，可能須移動 client 到別的 description 中的迴圈來取得所需的影格內容。為了避免破壞原迴圈的完整性，將清空已觀看過的 buffer 空間來讓原迴圈的 client 之間進行傳送。基於上述的理由，我們提出一個 buffer 管理機制，假設正觀看 buffer 為 now = 1 個時間單位，client buffer 空間為 sb，利用公式(1)、(2)將 sb，畫分成為 bf 和 bp 二區段，在 bf 中有一時間單位表示 client 正在觀賞的影片內容，其它為 bf，代表 client 將在未來會觀賞到的影片內容，而 bp 表示 client 已經觀賞後的影片空間。

$$bf = \left\lfloor \frac{sb-now}{2} \right\rfloor + now \quad (1)$$

$$bp = sb - bf \quad (2)$$

圖 5 client buffer 管理示意圖

如圖 5 為 client's Buffer 分配情況，假設中間那個空白方格代表一個單位時間的正觀看內容，斜線部份代表未觀看內容，網狀部份代表已觀看內容。



3. Inter-Description Recovery

Loopback-MDC 是假設於 CDN-P2P 架構下，利用 client 的資源，為其他 client 服務。server 無法預測 client 何時離開系統，因此當 client 中途離開系統時，造成迴圈斷裂發生時，在錯誤回復中，可分為內部串流描述修復機制和跨串流描述修復機制。

內部串流描述修復機(Intra-Description Recovery)

當 client 中途離開 server，因而影響接續在它之下的其他 client 正常觀看。在原本的 Loopback-MDC 機制中利用其他 client 的 buffer 中的重複的資料，傳送給受影響的 client，即可以使其他 client 依然保有正常的觀看。

跨串流描述修復機制(Inter-Description Recovery)

在原本的 Loopback-MDC 機制中，需要有其他 client 暫存重複的資料，才能進行內部串流描述修復機制，如果沒有任何 peer 有暫存其需要的資料，則需由 server 進行錯誤回復。所以我們提出了跨串流描述修復機制，可以利用別條 description 的資源，來修復 peer 的觀看品質，其中 Patching 修復法的前提為支援的 description 不可為需被修復 client 已持有的 description。例如：一部影片分為四條的 description，client x 請求三條 description，分別為 description 1、description 2、description 4，提供 description 2 給 client x 的 client (x-1) 中途離開系統，無法進行內部串流描述修復機制，client x 面臨缺少第 n 個影格，只能從 description 3 的迴圈當中，尋找有沒有第 n 個影格。

在跨串流描述修復機制中，可分二大修復步驟的方法： 縫合式修補(Sewing recovery)、補丁式修補(Patching recovery)。當錯誤發生時，依序進行修補方法，直到完成錯誤回復，以下進行這二種修復方法的詳細說明。

3.1 縫合式修復 (Sewing recovery)

當 description 中的 loop 將會發生斷裂時，無法利用內部串流描述修復機制時。在跨串流描述修復機制中先進行 Sewing 修復法，移動

其它 description 的 client 來修補這條斷裂迴圈，使迴圈能繼續完整串流傳送，不需要拆成二條迴圈，避免 server 再多提供一條上傳頻寬；且遺失的資料，無法被其它 client 修復時，暫由 server 進行修補，在 server 傳送完遺失的資料後，即可停止支援，以節省 server 頻寬。被移動的 client 其前提為不可造成原本迴圈發生斷裂。

time / des	N	N+1	N+2	N+3	N+4	N+5	N+6
d1		B-1			D-1		F-1
d2		B-2	C-1		D-2		
d3		B-3	C-2			E-1	
d4		B-4			D-3	E-2	

圖 6 未執行跨串流描述修復機制的 Sewing 的迴圈

time / des	N	N+1	N+2	N+3	N+4	N+5	N+6
d1		B-1					F-1
d2		B-2					
d3		B-3	C-2			E-1	
d4		B-4	C-1			E-2	

圖 7 執行跨串流描述修復機制中的 Sewing 後的迴圈

如圖 6、圖 7所示，假設每個client buffer為三個時間單位，因此當有三個時間單位沒有client銜接時，則將造成迴圈斷裂。每個client依其所需品質來決定它持有的 description數，如圖 6中client B請求full quality，即持有四條 description(B-1,B-2,B-3,B-4)。圖 6說明當client D中途離開系統時，d1 和d4 因為client D的離開而產生迴圈斷裂。因此，我們先進行Sewing修復，如仍未修補成功，則再進行Patching修復。如圖 7所示，利用Sewing修復把d2 中的C-1 加d4 的迴圈中，即把d2 中的C-1移入d4 的迴圈中，因為client C未持有 d4 的資料，因此不影響client C的正常觀看。但client C未持有d4 的資料，無法為d4 中的E-2 進行串流傳送，因此需要由server進行短暫的支援，為client E傳送d4 的資料，直到d4 中的C-1 從d4 中的B-4 接收到d4 的串流資料，則d4 中的C-1 可為d4 中的E-4 進行d4 的串流傳送，server便可停止支援。d2 中的C-1 可移動的原因在於它本身為最後一個節點。

3.1.1 縫合式 (Sewing) 修復優先權

為了使 Sewing 修復法更有效的降低 server 負載，本節討論 Sewing 修復優先權，首先尋找出開放式迴圈，並篩選出受影響最多的 client 的候選開放式迴圈，最後再挑選 client buffer 時間最長的候選開放式迴圈，來進行修復順序。

1. 找出開放式迴圈

在Sewing中，以開放式迴圈的description為優先修補的對象。因為這可以服務更多新進client。因此當有迴圈發生斷裂時，系統會找出受影響的description中為開放式迴圈的迴圈進行優先修補，若不只一條開放式迴圈發生斷裂，則考慮每個迴圈中受影響的client數量多寡，來來決定優先修補順序，圖 **8**為尋找開放式迴圈演算法。

2. 篩選受影響最多的 client 的候選迴圈

當有多條開放式迴圈受到影響，我們再從中挑選出最多受影響client數量的迴圈優先修復，可以避免這個迴圈斷裂後，會產生較多的錯誤client，使得server的負載大幅提高，如圖9為篩選出最多受影響client的迴圈演算法。

3. 挑選 client buffer 時間最長的候選迴圈

當有相同受影響總數的迴圈時，再挑選迴圈中最後client buffer時間最長的優先修復。避免剛修復完成的迴圈，因沒有較長的等待時間來等待新進client的加入，使得迴圈變成封閉式迴圈，而不能服務更多新進的client。如圖10為挑選迴圈中client buffer最長的演算法。如公式(3)，找出buffer time結束時間最晚的failed迴圈，假設EN為failed迴圈的集合，即需符合

$Max(T_{d,id}+ B_{d,i})$。$Peer_{d,i}$ 為description d 的第 i 個client；$T_{d,i}$ 為Description d 的第 i 個client 的 join time；$B_{d,i}$ 為Description d 的第 i 個client的 buffer time。

$$EN \leftarrow EN\{ d \mid Max(T_{d,id}+ B_{d,i}) ,1\le d\le m\} \quad (3)$$

```
sewing( FID, N ) //縫合式修補
 ON ← ∅; 發生failure的open loop
 R ← ∅; //已修復的description number集合
 PQ ← 0; // 受影響最多的peer數量
 EN ← ∅; // 候選待修補的loop
BN ← N; // 修補前的description number集合
ON ← ON ∪ {d | NowTime ≤ (T_{d,j_d} + B_{d,j_d})
                          , 1 ≤ d ≤ m, d ∈ N}
 repeat
      f( | N |= 1){  //只有一個failure loop,直接修補
            EN ← EN ∪ N;
            sew_recovery( EN );
     }else{
            //先補open loop
            if( | ON |= 0){  //都是close loop
                  EN ← EN ∪ {N ∉ ON};
                  max_failed_ peer( EN );
            }else if( | ON |= 1){  //只有一個open loop
                  EN ← EN ∪ ON;
                  sew_recovery( EN );
            }else {
             // 一條以上的open loop,需再選出受影響最多
                的peer 的loop
                  EN ← EN ∪ ON;
                  max_failed_ peer( EN );
            }
      }
 until  | N| = 0
```

圖 8 尋找開放式迴圈的演算法

```
/*找出受影響最多的peer 的failed oop*/
max_failed_ peer( EN ){
        temp ← 0;
        q ←| EN |;
        tempN ← EN;
        for ( n =1 ,n ≤ q, n++){
                i ← 0;
                依序從tempN取出d;
                tempN ← tempN − {d};
                tempP ← P_d ;
                for( k = 1, k ≤ j_d , k++ ){
                      依序從tempP取出PID;
                      tempP ← tempP − {d};
                      if( PID = FID )
                      //尋找description d 中
                          FID peer 的順位
                       i ← k;
                      break;
                }
                / /受影響的peer數量
                temp = j_d − i;
                 if ( temp > PQ){
                        PQ ← temp;
                        / /最受影響的peer較多的
                            loop
                        EN ← d;
                / /受影響的peer數量相同
                }else if( temp=PQ){
                        / /候選待修補的loop
                        EN ← EN ∪ d;
                }
          }
        //有一個以上待修補的open loop
         if( | EN| >1 && | ON| >1)
              last_buffer_time( EN );
}
```

圖 9 篩選受影響最多的 client 的候選開放式迴圈演算法

```
/*找出buffer time結束時間較晚的failed loop*/
 last_buffer_time( EN ){
     EN ← EN ∪ { d | Max(T_{d,j_d} + B_{d,j_d}) , 1 ≤ d
     ≤ m , d ∈ EN};
}
```

圖 10 挑選 client buffer 時間最長的候選開放式迴圈的演算法

3.2 補丁式修復 (Patching recovery)

當進行 Sewing 修復後，如果還有 client

無法順利的進行觀看影片時，再使用 Patching 修復法替受影響的 client 找尋其它可加入且 client 尚未持有的 description 的迴圈中，且可加入的 description。

time / des	N	N+1	N+2	N+3	N+4	N+5	N+6
d1		B-1					F-1
d2		B-2					
d3		B-3	C-2			E-1	
d4		B-4	C-1			E-2	

圖 11 未執行跨串流描述修復機制中 Patching 的迴圈

time / des	N	N+1	N+2	N+3	N+4	N+5	N+6
d1		B-1					
d2		B-2					
d3		B-3	C-2			E-1	F-1
d4		B-4	C-1			E-2	

圖 12 執行跨串流描述修復機制中 Patching 後的迴圈

如圖 11所示，當進行Sewing完後，還有一個d4 未成功修補，再使用Patching修復。利用Patching修復把d1 中的F-1 加d3 的迴圈中，即把client F中d1 中的F-1 移d3 的迴圈中，如圖12。

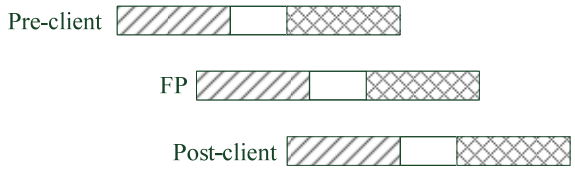


圖 13 Patching 修復運作方法

圖 13為FP插入到別description。Pre-client和Post-client為description x中的迴圈，FP為description y因發生迴圈產生斷裂而產生的錯誤client，而FP所需的影片可加入description x的迴圈內的Pre-client來取得，所以搬移FP到description x中的迴圈時。FP利用bf來降低FP進行搬移所產生的影片延遲，而且description x 迴圈中的Pre-client和Post-client因FP的加入，可能會發生傳輸影格錯誤。如圖 14所示，假設每個client的buffer長度為 5，其bf為 3，bp為 2，而FP為別條description發生斷裂後產生的錯誤client，為了修復此client，將FP加入別的description有它需要的資料的迴圈中。如圖14 (a)所示，FP插入到別的迴圈中的p1(Pre-client)和p2(Post-client)之間，FP清空bp區段，讓p1 和p2 來使用它，經過一單位時間後，如圖 14(b)所示，p1 傳送影格 26 給FP再給p2。再經過一單位時間後，如圖 14 (c) 所示，p1 傳送影格 27 給FP再給p2，最後在圖 14 (d) 所示，雖然FP可以從P1 得到所需的影格29，但是FP無法傳送影格 28 給P2，因為FP所持有的資料是description y的影格 28，於是P3將無法順利觀看影片。

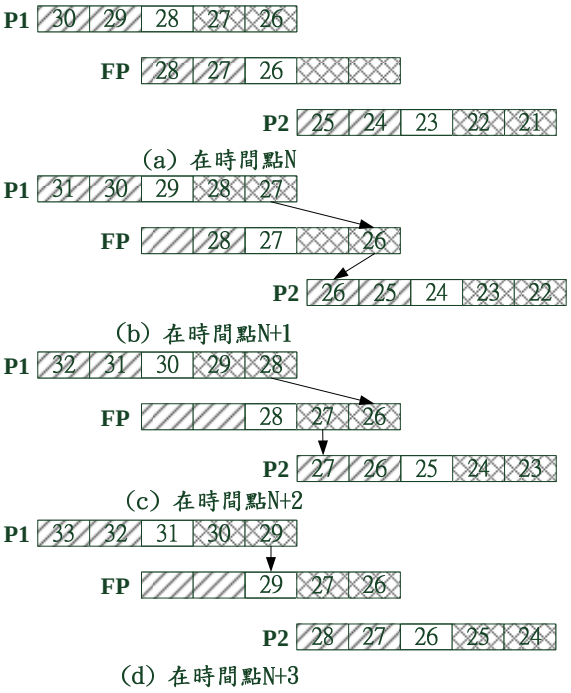


圖 14 Patching 執行失敗範例

3.2 補丁式修復 (Patching) 條件

為了避免發生 Patching 修復錯誤，我們必須限制 FP 進入 Pre-client 和 Post-client 的區間，避免因 FP 加入使得原本迴圈發生斷裂。為了 Patching 修復能順利完成修補，我們必須

考慮 FP 加入的時間點，是否會影響原有迴圈的傳遞結構，在我們深入研究後，須符合以下三個條件：

1. FP 加入時間小於或等於 Pre-client buffer 結束時間

確保 Pre-client 可以提供 FP 後面所需的影格內容。如公式(4)，找出滿足 FP 加入時間小於或等於 Pre-client buffer 結束時間的 $Peer_{d,i}$。$Peer_{d,i}$ 為 description d 的第 i 個 client；$T_{d,i}$ 為 Description d 的第 i 個 client 的 join time；$B_{d,i}$ 為 Description d 的第 i 個 client 的 buffer time。$T_{fd,fi}$ 為發生錯誤 Description d 的第 i 個錯誤 client 的 join time；

$$T_{fd,fi} \le T_{d,i} + B_{d,i} \quad (4)$$

2. FP 中 bf 最大的影格編號必須大於或等於 Post-client bf 中最大的影格編號

需考慮 FP 加入系統時間以及插入的位置，確保迴圈的排列能使 client 間順利傳送資料，避免原本迴圈因 FP 的加入而發生迴圈斷裂。如公式(5)，找出符合 FP 中 bf 最大的影格編號必須大於等於 Post-client bf 中最大的影格編號的 $Peer_{d,i}$。$\alpha_{d,i}$ 為 description d 的第 i 個 client 未觀看 buffer 中最大的影格編號；$\alpha_{fd,fi}$ 為發生錯誤 description d 的第 i 個錯誤 client 未觀看 buffer 中最大的影格編號。

$$\alpha_{d,i} \le \alpha_{fd,fi} \le \alpha_{d,j+i} \quad (5)$$

3. FP 中 bf 最小的影格編號必須小於或等於 Post-client bf 中最大的影格編號

為確保在進行修補時，FP 有足夠的未觀看影格進行緩衝修補時間。如公式(6)，找出符合 FP 中 bf 最小的影格編號必須小於等於 Post-client bf 中最大的影格編號的 $Peer_{d,i}$。即需符合 $\beta_{fd,fi} \le \alpha_{d,j+I}$。$\alpha_{d,i}$ 為 description d 的第 i 個 client 未觀看 buffer 中最大的影格編號。$\beta_{fd,fi}$ 為發生錯誤 description d 的第 i 個錯誤 client 未觀看 buffer 中最小的影格編號。

$$\beta_{fd,fi} \le \alpha_{d,j+I} \quad (6)$$

圖 15 為滿足 Patching 修復條件的演算法，找出滿足 $T_{fd,d,i} \le T_{d,i} + B_{d,I}$、$\alpha_{d,i} \le \alpha_{fd,fi} \le \alpha_{d,j+i}$ 和 $\beta_{fd,fi} \le \alpha_{d,j+I}$ 這三個條件的 $Peer_{d,i}$。

```
patching( FID, N ) //補丁式修復
  repeat
      Peer_{d,i} ← 0;
      Peer_{d,i} | T_{fd,fi} ≤ T_{d,i} + B_{d,i} , / / pro − peer
          α_{d,i} ≤ α_{fd,fi} ≤ α_{d,i+1}, / / post − peer
          β_{fd,fi} ≤ α_{d,i+1}, Peer_{d,i} ∉ CX,
          d ∉ N, d ∈ O, d ∉ H_{fd,fi};
          EX ← EX - {Peer_{fd,fi}};
          if (Peer_{d,i} > 0) / /找到可以進行修復的peer
              patch_recovery();
  until | EN| = 0
```

圖 15 Patching 演算法

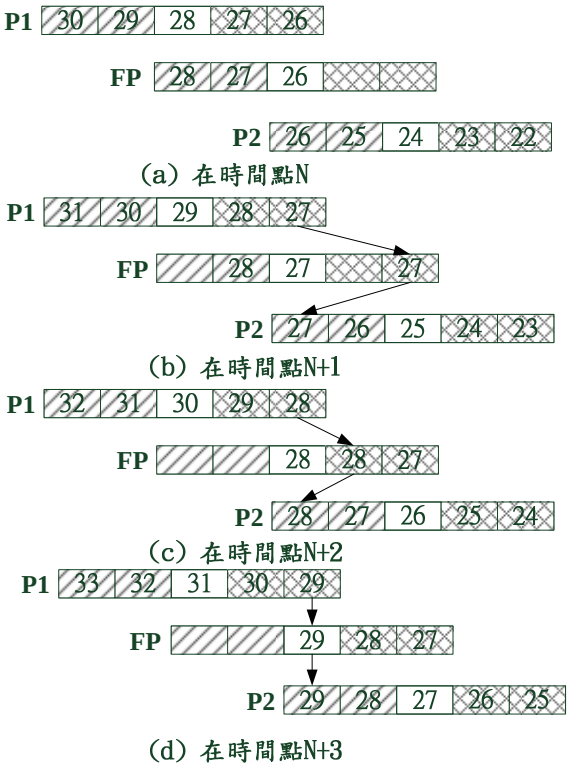


圖 16 Patching 執行成功範例

Patching 修復成功範例，在圖 16(a)FP 加

入 p1 和 p2 之間已經先滿足前述三個條件。所以當三個時間單位後，在圖 16(d) p2 順利修補完影格 28 後，剛好接上 FP 正需接收影格 29，順利完成補丁式修復。

4. 連續離開的 client 之問題探討

當迴圈產生斷裂時，在 Loopback-MDC 中，如果無法利用內部串流描述修復機制來修復此 client 時，將由 server 來提供其 client 所需的 description。

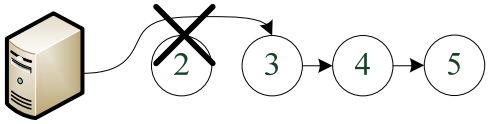


圖 17 單一 離開的 client

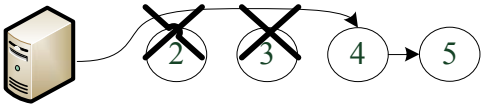


圖 18 連續離開的 client

如圖 17所示，假設編號為 3 的client因前面有client中途離開，導致迴圈斷裂，經過修復後，仍然無法銜接起來，此時由server提供所需的description，來服務編號為 3 的client。但不是每個發生錯誤的client都需要由server提供額外的上傳頻寬來進行修補。如圖 18所示，由server提供影格內容的編號為 3 的client中途離開，導致編號為 4 的client發生斷裂。而編號為 3 的client是由server 所提供其影片內容，所以在編號為 3 的client離開，server將原先上傳到編號為 3 的client中description轉送到編號為 4 的client，所以對server而言，並未因編號為 3 的client中途離開，而再度提供其上傳頻寬。

為了方便觀察此現象，我們定義一個集中率 σ，其計算為公式(7)，發生迴圈斷裂的 client 的總數量為 n，其產生連續區塊的總數為 λ。

$$\sigma = \frac{n-\lambda}{n-1} \qquad (7)$$

假設總共有 10 個 client 中途離開迴圈，而這些 client 有 7 個 client 在離開後，會造成迴圈斷裂，如圖 19 所示，因中途離開會產生斷裂的 client 編號為 2、3、4、8、9、12、15 時，則 n 為 7、λ 為 4、σ 約為 0.5。其 σ 越接近於 1，代表斷裂的 client 中的連續丟棄的情況越多， server 須提供的上傳頻寬反而可能會降低。

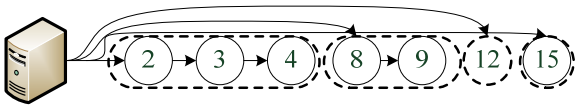


圖 19 server 支援 client 示意圖

5. 效能評估與分析

利用本研究所提的跨串流描述修復機制與 Loopback-MDC 內部串流描述修復機制進行模擬實驗，實驗中設定影片分為 16 條 description；模擬時間為 2000 個單位時間；client 請求的 description 數量，依照常態分配 (μ =5，σ =3) ；丟棄方法為隨機丟棄；丟棄數量為全部 client 數乘以丟棄率；評估這二種修復方法在不同丟棄率的變化下，離開的 client 分佈、server 負載和修復 client 的成功率。

4.1 離開的 client 分佈情況

以下為隨機丟棄client後，所進行離開client分佈情況模擬實驗。使用者buffer長度為 3 單位時間。評估在不同的丟棄率下client連續區塊的情況。如圖 20在丟棄率 10%、20%、30%、40%、50%區段，集中率 σ 偏低；在 40%和 50%這區段，可發現 50%亂數丟棄的client產生的連續區域並沒有明顯上升並低於 0.5，代表其分散client偏多；在60%和70%區段中，集中率 σ 在 0.7 以上，表示發生斷裂的client的連續區塊增多。

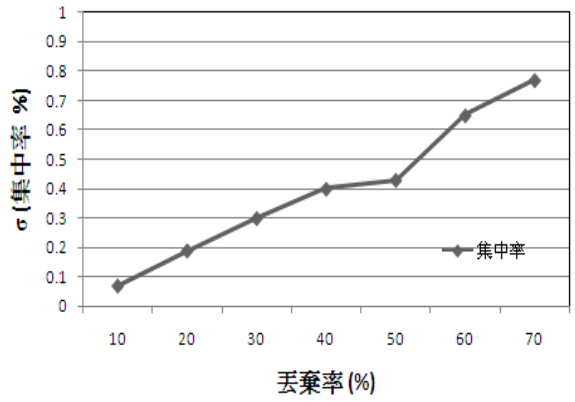


圖 20 不同丟棄率之集中率的比較

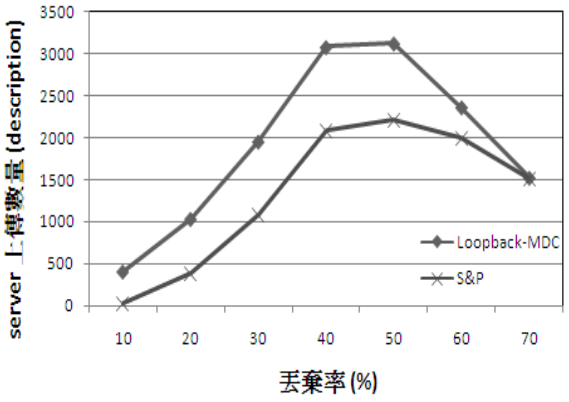


圖 21 不同丟棄率之 proxy server 上傳頻寬的比較

4.2 server 頻寬耗用

以下以跨串流描述修復機制和 Loopback-MDC 內部串流描述修復機制進行模擬實驗。使用者 buffer 長度為 3 單位時間。評估二種分配方法在不同的丟棄率下 server 上傳頻寬。

如圖 21所示 ，在丟棄率 10%下完全藉由client之間來互相提供， server不需要額外提供上傳頻寬;在丟棄率 50%中不論跨串流描述修復機制和Loopback-MDC，這二者的丟棄client已經達到50%了,卻只呈現出趨緩上升，而沒有大幅提高server負載。這情況代表前小節敘述情況在丟棄率高下已經影響到server負載。 但如圖 21所示，σ 在丟棄率 50%時小於 0.5，這代表在丟棄率 50%中所丟棄client會造成迴圈斷裂的連續client不多，使server負載仍然會上升，但呈現趨緩；在丟棄率 60%和 70%中在 σ 達到 0.7 以上，這代表在丟棄率 60%和 70%中所丟棄client會造成迴圈斷裂的連續client居多，導致server負載不在上升，反而因服務的client變少而降低；在丟棄率 70%中，跨串流描述修復機制和Loopback-MDC內部修復因client太少，所以無法成功修補，全由server來進行服務。整體而言，使用跨串流描述修復機制搭配Loopback-MDC將可以更有效的降低server的上傳頻寬。

4.3 修補成功率

以下為跨串流描述修復機制和 Loopback-MDC 內部修復法，再評估這二種分配方法於不同的丟棄率下整體的堅韌度。在迴圈中當有 client 一離開即造成迴圈斷裂，且無法利用 buffer 中重複的串流資料進行內部修復,稱這些的 client 為 fragility (脆弱性) client。我們定義一個成功率 α ，其計算為公式(8)。經修復後仍由 client 服務的 client 總數量為 κ，fragility client 總數量為 τ。當成功率 α 越高則代表被修復的節點越多。

$$\alpha = \frac{\tau}{\kappa} \qquad (8)$$

如圖 22所示，跨串流描述修復機制的成功率 α 比Loopback-MDC內部串流描述修復機制來得高。因為在Loopback-MDC中的內部修復法只能利用其他的client 的buffer中的重複的資料,傳送給受影響的client。如圖 23所示，在丟棄率 10%中，S&P完全不受丟棄client的影響，而Loopback-MDC內部修補出現 20%的client數量需要由server提供服務；在丟棄率 50%以上，因迴圈中的離開的client過多，導致

修補成功率 α 大幅降低;在丟棄率達到 70%，因丟棄client過多，所以迴圈中的client無法為下個client服務，只能全部由server提供所需的內容，導致成功率 α 為 0。整體而言，S&P在丟棄率 40%以下，可以更有效利用迴圈 中的 client 來為新進client 提供更多的服務，大部分的請求都可由client之間來進行服務。

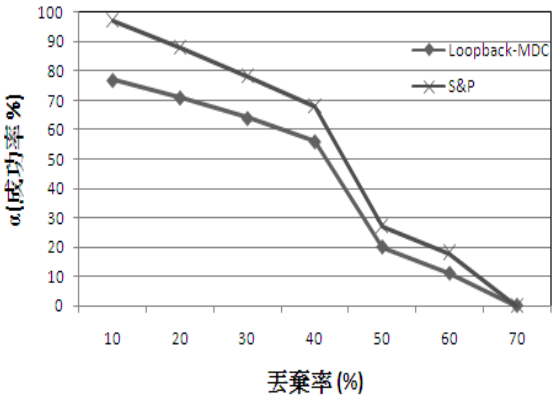


圖 22 不同丟棄率之成功率的比較

6. 結論與未來研究

本研究主要提出新的跨串流描述修復機制，來強化 Loopback-MDC 中的內部串流描述修復機制。使得 Loopback-MDC 機制在 client 中途離開時，受影響的 client 可以由其它的 client 來進行修復，維持 client 所需的觀看品質，降低 server 在迴圈斷裂的時候，所付出的補償上傳頻寬。經由模擬實驗結果證實，S&P 方法可以更有效加強迴圈堅韌度策略，讓中途離開的使用者，不會影響其他使用者的觀看，讓提供的串流服務品質更加穩定。主要是因為 Sewing 主要修復原則為維持迴圈為開啟的狀態，讓修復完的迴圈可以服務更多新進client。而 Sewing 未能修復的 client 可以透過 Patching 在不影響下其它迴圈下提供所需的內容，因此跨串流描述修復機制可以有較效率果。

目前研究中，運用新的錯誤回復技術，假設 client buffer 為固定值，而實際上每個 client buffer 大小應當有所不同，所以當發生新加入的 client buffer 結束時間遠小於最後一個 client buffer 結束時間，會降低迴圈的可用性。在未來研究中，我們將探討當 client 的 buffer 大小不一致時，如何利用 client buffer 來延長迴圈為開啟的狀態，提高整體迴圈的可用性，以降低 server 須額外提供上傳頻寬的可能性。

7. 參考文獻

[1] 林朝興, 李宜婷, " Loopback-MDC: 在 CDN-P2P 網路上利用支援多重編碼描述的協力式快取提升影音串流之延展 , " ***2007 全國計算機會議 (National Computer Symposium, NCS)*** ,pp.700-711, 2007。

[2] A. Dan , D. Sitaram and P. Shahabuddin, “Scheduling policies for an on-demand video server with batching,” ***Proc. of ACM MM***, pp.15-23, 1994.

[3] C.-L. Chan, S.-Y. Huang, H.-H. Chen, W.-H. Tung, and J.-S. Wang, "An application-level multicast framework for large scale VOD services," ***Proc. of 11th International Conference on Parallel and Distributed Systems***, Vol. 1, pp.98-104, 2005.

[4] C.-L. Chan, S.-Y. Huang, N.-C. Lin, J.-S. Wang, "Performance analysis of caching strategies for proxy-assisted VOD services," ***Information Technology and Applications, Proc. of Third International Conference on ICITA***,Vol. 2, pp.387-392, 2005.

[5] D. A. Tran, K. A. Hua, and T. Do, “ZIGZAG: An efficient peer-to-peer scheme for media streaming,” ***Proc. of IEEE INFOCOM, Vol. 2***, pp.1283-1292, 2003.

[6] E. Kusmierek, Y. Dong and D. H. C. Du, “Loopback: Exploiting collaborative caches for large-scale streaming,” ***IEEE Trans. on Multimedia***, Vol. 8, no. 2, pp.233-242, 2006.

[7] J. G. Apostolopoulos and S. J. Wee , “Unbalanced multiple description video communication using path diversity,” ***Proc. of IEEE ICIP***, Vol. 1, pp.966-969, 2001.

[8] K. A. Hua, Y. Cai, and S. Sheu, "Patching: a multicast technique for true video-on-demand services," ***Proc. of ACM MM***, pp.191-200, 1998.

[9] T.-C. Su, S.-Y. Huang, C.-L. Chan, and J.-S. Wang, “Optimal chaining scheme for video-on-demand applications on collaborative networks,” ***IEEE Trans. On Multimedia***, Vol. 7, no. 5, pp.972-980, 2005.

[10] V. K. Goyal, “Multiple description coding: compression meets the network,” ***Signal Processing Magazine of IEEE***, Vol. 18, pp.74-93, 2001.

[11] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," ***Proc. of ACM NOSSDAV***, pp.177-186, 2002.

[12] Y. Dong, E. Kusmierek, and Z. Duan, “Exploiting limited upstream bandwidth in peer-to-peer streaming,” ***Proc. of IEEE ICME***, pp.1230-1233, 2005.

[13] Y. Wand, A. R. Reibman, and S. Lin, “Multiple description coding for video delivery,” ***Proc. of the IEEE***, pp.57-70, 2005.

[14] Y.-H. Chu , S. G. Rao, S. Seshan, and H. Zhang, “Enabling conferencing applications on the Internet using an overlay multicast architecture,” ***Proc. of ACM SIGCOMM***, pp.55-67, 2001.

[15] Z. Lu and W. A. Pearlman, “An efficient, low-complexity audio coder delivering multiple levels of quality for interactive application,” ***Proc. of IEEE Multimedia Signal***, pp.529-534, 1998