在 CDN-P2P 上使用 ExtLoopback-MDC 提升影音服務的可用性

林朝興	陳安琪	林哲維	李明憲
資訊工程系	資訊工程系	資訊工程系	資訊管理系
國立台南大學	國立台南大學	國立台南大學	南台科技大學
e-mail：	e-mail：	e-mail：	e-mail：
mikelin@mail.nutn.edu.tw	pal2339@hotmail.com	sh900010@hotmail.com	m9690237@webmail.stut.edu.tw

摘要 — 隨著網路的頻寬快速增加，更多使用者透過隨選視訊伺服器直接點選喜歡的影片進行觀看。如何有效率、大範圍地提供多媒體影音串流服務，是一項值得關注的重要議題。本研究針對 Loopback-MDC 架構，提出 ExtLoopback-MDC 的方法。ExtLoopback-MDC 架構是用來改善因客戶端到達率偏低，造成相距太遠而無法互相分享影音串流，導致代理伺服器負載提高的情況。透過此架構，有效的提升影音服務的可用性而達到降低代理伺服器的頻寬耗損。模擬實驗結果顯示也顯示，相對於以往的 Loopback-MDC 架構，使用 ExtLoopback-MDC 來提供影音串流，隨選視訊代理伺服器可以用較少的頻寬來服務各種不同到達率、失敗率的使用者。

關鍵詞: 串流，錯誤回復，同儕網路，多重描述串流編碼，代理伺服器

一、前言

隨選視訊(Video-on-Demand, Vod)伺服器 主要在提高系統的延展性 (scalability)，每當大量的客戶端觀看請求到達(flash crowd) 或離開時，伺服器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。傳統的客戶端-伺服器架構，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[3]、Patching[9]等技術，但上述所提到的數種作法皆建立在 IP multicast。但在目前已存在的網路架構上，實作並不容易。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代理伺服器，並將影片內容放置於中，藉由這些伺服器來分擔單一伺服器的負載，避免無法負荷，但若要增加延展性，則須大量提高成本與資源。而後來的 Peer-to-Peer(P2P)[6][15]架構運用客戶端為服務端亦為接受端的特性，使客戶端暫存空間中的影片內容，可以與其它客戶端進行影片傳送，支援伺服器固定的頻寬，進一步降低伺服器的負載。當大量的客戶端請求出現，可以提供大量的頻寬來支援現有的伺服器頻寬，解決大量請求的問題，並提高 VoD 伺服器的延展性，但是 P2P 的劣勢在於客戶端的加入以及離開，並非伺服器可以控管。因應上述的問題，有學者提出 Loopback [7]機制應用在 CDN-P2P[4][5][13]混合架構上，利用了現有的架構，並排除其缺點，使其具有高擴充性。但在 Loopback 機制上，只提供了單一串流來服務客戶端，並且客戶端之間存在網路異質性的問題，造成伺服器要把影片設定為最低的串流品質，才能讓大部份的客戶端互相順利傳遞串流。基於上述的原因，有學者後續進一步提出 Loopback-MDC[1][2]機制，在 Loopback 中利用

* 本研究承蒙國科會部分贊助，計畫編號：NSC 97-2221-024-014-MY3

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。

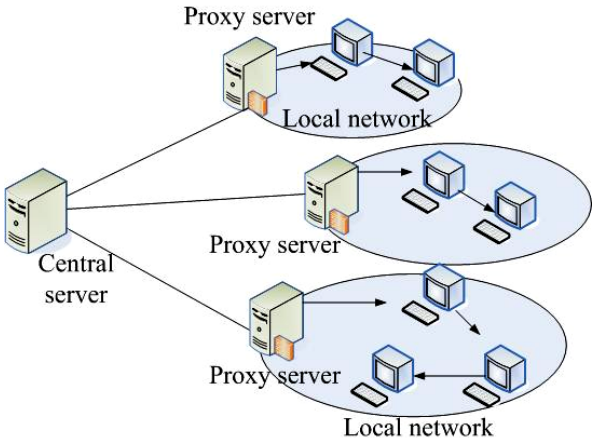


圖 1 CDN-P2P 混合架構

在 CDN-P2P 中有中央伺服器(Central Server)及多台代理伺服器(Proxy Server)，如圖 1。中央伺服器存放所有的影片來源，並提供代理伺服器影片內容以及支援代理伺服器來給予客戶端影片內容；代理伺服器建構在各區域中，並負責 cache 比較熱門的影片，提供區域下的客戶端進行影片傳送的服務。若客戶端所請求的影片，代理伺服器已持有，則可直接由它來服務。代理伺服器持有的熱門影片，可以提供大多數客戶端所需的服務，進而降低中央伺服器的負載程度。透過同區域和 P2P 的特性，使得客戶端可以互相支援，讓影片內容可以更快速的進行分享並且穩定的傳送內容。當有多個客戶端觀看同一部影片時，由最早觀看的客戶端來跟代理伺服器取得影片內容的片頭，再由客戶端之間進行服務，不需要再經由代理伺服器或中央 伺服器來服務，進而可以提高服務的數量。

Loopback-MDC 採用 CDN-P2P 混合架構，假設進來觀看影片的客戶端都擁有暫存影片內容和動態暫存影片內容於 buffer 中的能力，因此客戶端能利用 buffer 中的暫存內容進行串流傳輸給其它客戶端，但客戶端在進行傳送時，可能會失敗或是中途離開伺服器。

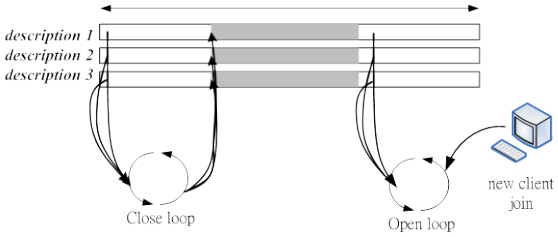


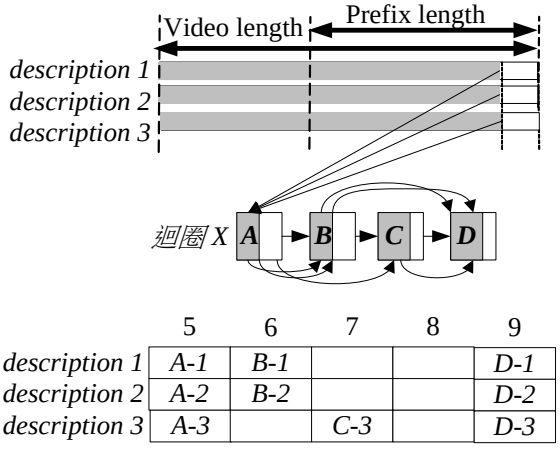
圖 2 Loopback-MDC 示意圖

每個代理伺服器 建立在區域網路上，並且暫存較熱門影片的前半段影片內容，來服務同區網中的客戶端。在相同區域的客戶端觀看一樣的影片時，會依進入的時間，循序產生數個迴圈。當客戶端送出請求時，代理伺服器 提供串流服務，但當代理伺服器 無法滿足客戶端的請求，又或者客戶端已經觀看到影片片尾時，則由中央 伺服器 來提供所需的影片內容。在代理伺服器中的影片內容透過 MDC 編碼機制[8][11][12][14]，產生多條的 description，如圖 3。客戶端可依其請求來接收 description 數量，即客戶端接受的 description 數量越多，表示其觀看品質越高；接受到全部的 description 數量，表示此客戶端所觀看的影片為最佳品質(full quality)。當第一個進來的客戶端根據自身所須的請求從代理伺服器 取得數條 description 的影片片頭來進行觀看，在進行一段時間後，將可服務後來加入迴圈中的客戶端。當影片片頭一直保持在客戶端之內，則表示可以不斷的服務新加入的客戶端，如圖 3 中的開放式迴圈。因每個客戶端所要求的 description 數量不同，並且為了確保影片片頭能夠長時間暫存於客戶端的儲存空間中來服務更多的新進客戶端，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(Open Loop First, OLF) Description 分配。

OLF description 分配法，主要在進

description 分配時，OLF 先找出為開放式迴圈的 description，再以這些 desc*ription* 進行分配讓這些迴圈維持開啟的狀態。當新進客戶端所需的 description 數量高於現有的開放式迴圈，則由伺服器開啟新的開放式迴圈來滿足新進客戶端的請求，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。其中，每一個新開啟的迴圈，都需耗用一條伺服器的上傳頻寬。

當開放式迴圈因最後一個客戶端的儲存空間填滿，且沒有再出現相同請求的客戶端而變成封閉式迴圈，最後一個客戶端會把最後的影片片段轉送到代理伺服器，若代理伺服器的 buffer 夠大，則後續在觀看到影片尾的時候，也不用跟中央 伺服器請求，而可以直接由代理伺服器來進行服務。



	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 3 Loopback-MDC 之 OLF 分配

如圖 3 所示，假設代理 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,，每個客戶端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，就可以加入迴圈 X。在迴圈 X 中，有 *Client A* 請求三條 description、*Client B* 請求二條 description 及 *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的 description，不需由伺服器 提供新的 description。

論文章節安排如下，在第二節，探討相關研究的文獻；第三節探討如何執行 ExtLoopback-MDC 機制；第四節探討 ExtLoopback-MDC 的問題;第五節描述研究的實驗結果與分析；第六節為結論與未來研究。

二、文獻探討

Loopback-MDC [2]是於 CDN-P2P 架構下，利用客戶端的資源，為其他客戶端服務。伺服器無法預測客戶端何時離開系統，因此當客戶端中途離開系統而造成迴圈斷裂發生時，就必須要錯誤修復的機制。而在錯誤修復的機制中，又可分為內部串流描述修復機制和跨串流描述修復機制。

內部串流描述修機制(Intra-Description Recovery)

當客戶端中途離開伺服器，因而影響接續在它之下的其他客戶端正常觀看。在原本的 Loopback-MDC 機制中，利用其他客戶端的 buffer 中重複的影格資料，傳送給受影響的客戶端，即可使其他客戶端依然可以正常的觀看。

跨串流描述修復機制(Inter-Description Recovery)

利用別條 description 的資源，來修復 peer 的觀看品質，此類修復法的前提為支援的

description 不可為需被修復客戶端已持有的 description。例如：一部影片分為四條的 description，客戶端 *x* 請求三條 description，分別為 *description 1*、*description 2*、*description 4*，提供 *description 2* 給客戶端 *x* 的客戶端 (*x-1*)中途離開系統，無法進行內部串流描述修復機制，客戶端 *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。

當 description 中的 loop 發生斷裂而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[1]，移動其它 description 的客戶端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，避免伺服器再多提供一條上傳頻寬；且當遺失的資料無法被其它客戶端修復時，暫由伺服器進行修補。在伺服器傳送完遺失的資料後，即可停止支援，以節省伺服器頻寬，而被移動的客戶端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				
d3		B-3	C-2		D-2	E-1	
d4		B-4			D-3	E-2	

圖 4 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					
d2		B-2					F-1
d3		B-3	C-2		E-1		
d4		B-4	C-1		E-2		

圖 5 執行跨串流描述修復機制中的 Sewing 後的迴圈

如圖 4、圖 5 所示，假設每個客戶端 buffer 為三個時間單位，因此當有三個時間單位沒有客戶端銜接時，則將造成迴圈斷裂。每個客戶端依其所需品質來決定它持有的 description 數，如圖 4 中 *Client B* 請求最佳品質，即持有四條 description(*B-1,B-2,B-3,B-4*)。圖 4 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 5 所示，利用 Sewing 修復把 *d2* 中的 *C-1* 加入 *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，因此需要由伺服器進行短暫的支援，替 *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-2* 進行 *d4* 的串流傳送，伺服器便可停止支援。*d2* 中的 *C-1* 可移動的原因在於它本身為最後一個節點。

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 6 執行跨串流描述修復機制中 Patching 後的迴圈

當進行 Sewing 修復後，如果還有客戶端無法順利的進行觀看影片時，再使用 Patching 修復法[1]替受影響的客戶端 找尋其它可加入且本身尚未持有的 description 加入。如圖 4 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，即把客戶端 *F* 中 *d1* 中的 *F-1* 移 *d3* 的迴圈中，如圖 6。

三、ExtLoopback-MDC

在這個章節，我們將針對客戶端的可用性來進行探討。在 P2P 網路上提供隨選視訊的服務中，客戶端的儲存空間大小，決定了一個客戶端的可用性(availability)。一個客戶端的儲存空間愈大， 表示該客戶端能夠儲存更長時間的影片內容且提供較長時間的影片串流服務給其他客戶端。因此，在 P2P 網路上具有較大儲存空間的客戶端，將會具有較高的可用性，反之則愈低。在 Loopback-MDC 中客戶端同樣也具有上述的特性，如果迴圈中的客戶端能夠把影片片頭在儲存空間中儲存越久，開放式迴圈的狀態也能持續較長的時間。因此新進客戶端可以直接的從開放式迴圈中取得影片內容，不需再向伺服器端請求，所以提升客戶端最佳的利用率,將會提升系統的效能，因此我們提出了 Extloopback-MDC 機制。

3.1 Extloopback-MDC 機制

當新加入客戶端儲存空間的結束時間遠遠小於最後一個客戶端儲存空間的結束時間， 則把新加入客戶端和最後一個客戶端的串流順序交換，可拉長整個迴圈等待下一個客戶端的時間。

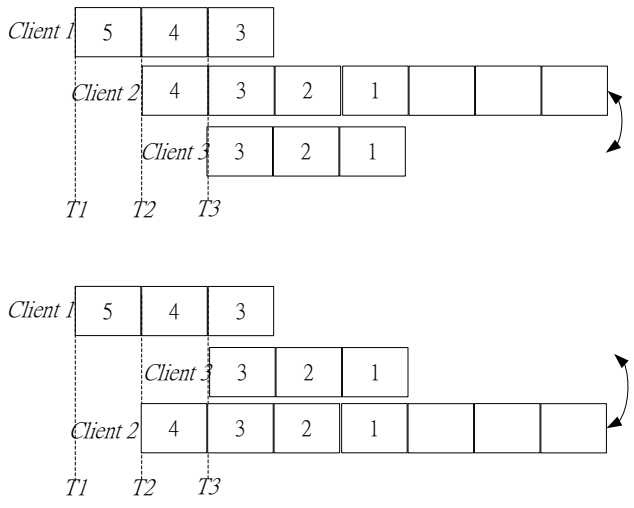


圖 7 調整客戶端傳輸順序示意圖

如圖 7，左邊為原始客戶端加入伺服器的順序。發現 *Client 2* 雖然比 *Client 3* 先加入 ，但 *Client 2* 儲存空間的長度比 *Client 3* 儲存空間的更長，為了延長客戶端 buffer 等待下一個客戶端加入的時間，於是調整客戶端的傳輸次序，把 *Client 3* 和 *Client 2* 進行傳輸次序對調，再進行傳輸。

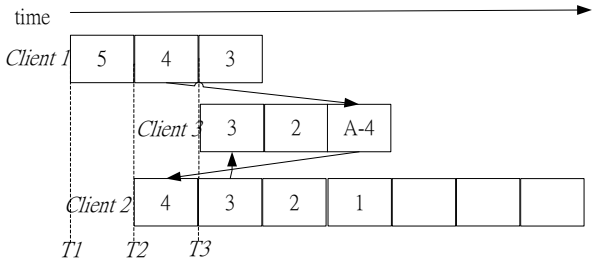


圖 8 調整客戶端傳輸順序示意圖

如圖 8 中 *Client 3* 需同時接收 *Client 1* 和 *Client 2* 並且 *Client 2* 需傳送 *Client 3* 尚未觀看的串流，最後再由 *Client 3* 從 *Client 1* 接收的資料傳給客戶端 2，即完成調整。

表 1 中為每個客戶端進來的時間與 buffer 大小，如圖 9 所示，在 *time* = 3 時，迴圈的狀態為開放式迴圈且內有 *Client A* 、*Client B* 及 *Client C* 依原本 loopback-MDC 串連起來，而在 *time* = 5 時， *Client C* 的 buffer 已滿，而 *Client D* 尚未加入， 所以在下一時間點迴圈的狀態將變為關閉式迴圈。因為片頭 1 已經不存在，這時候將使用 ExtLoopback-MDC 機制，透過 *Client B* , *Client C* 來計算，*Client B* 可以延長較多的等待時間，所以我們在 *time* =6，進行 Extloopback-MDC 機制， 可以看到透過 Extloopback-MDC 機制過後的影片傳遞情況；*Client B* 透過 *Client C* 向 *Client A* 拿到所需的影格內容， 在 *time* = 6 中 *Client A* 需多負擔一條上傳頻寬， 而在 *time* = 7 中 *Client A* 不需額外負擔，表示經過 Extloopback-MDC 機制後， *Client* 只需短暫提供額外的上傳頻寬， 而客戶端 *B* 可以使整個迴圈的開放式狀態延至 *time* = 8 ， 並順利在 *Client D* 進來時，提供了片頭 1。這使得伺服器不需要額外開起新的迴圈，有效降低伺服器的負載。

表 1 客戶端進來時間與 buffer

	A	*B*	*C*	*D*
進來時間	1	2	3	8
儲存空間	5	7	3	3

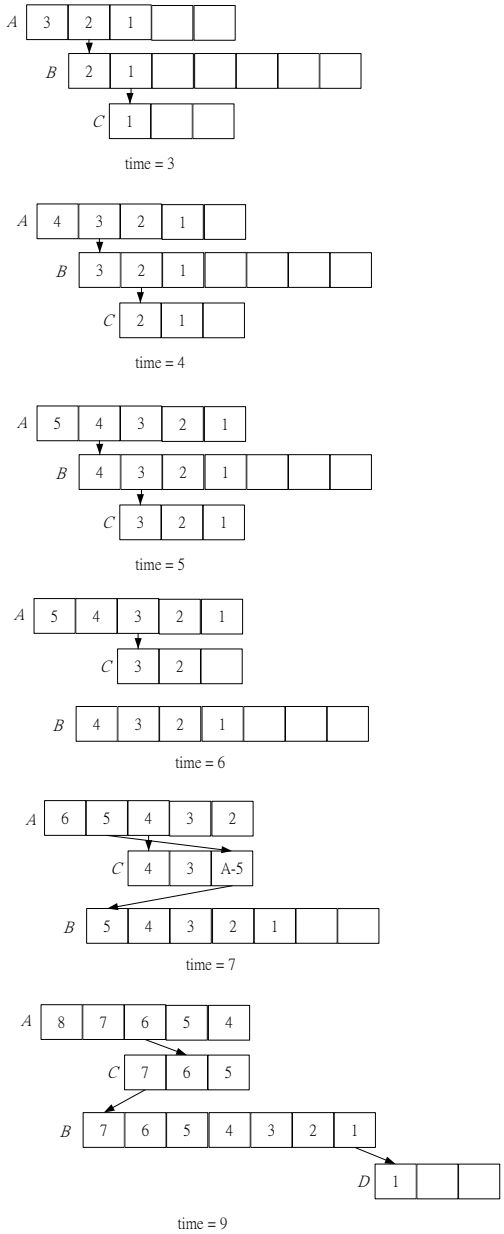


圖 9 ExtLoopback-MDC 運行成功範例

3.2 ExtLoopback-MDC 執行條件

為了降低 Extloopback-MDC 機制的複雜度並避免發生客戶端長時間提供額外的上傳頻寬，客戶端進行的 Extloopback-MDC 機制行為必須有所限制。為了使 Extloopback-MDC 機制能順利達到要求，我們必須考慮使 ExtLoopback- MDC 機制的時間和客戶端儲存空間的大小，是否會影響原有迴圈的傳遞結構，導致客戶端可能會需要長時間支付額外的頻寬並使得影片傳遞的複雜度大幅提高。ExtLoopback-MDC 執行時，須符合以下三個條件：

條件1

迴圈最後一個客戶端的儲存空間已滿，而新進客戶端尚未到來

儲存空間未滿表示還有等待的時間，不需要增加不必要的複雜度來延長整個迴圈的開放式狀態，如圖 10 所示，*Client A* 在沒有必需的情況下，多增加了一條上傳頻寬。

條件2

迴圈最後一個客戶端較之前的客戶端有比較長的等待時間

客戶端替換的儲存空間沒有較原來長的等待時間，那運行機制後，反而對整條迴圈沒有幫助，一樣由 伺服器提供額外的上傳頻寬，如圖 11 所示，經過交換後，較短的 *Client B* 的儲存空間成為可以等待新加入客戶端的時間，比不進行交換前的 *Client B* 少了 6 個時間單位。

條件3

(被交換的客戶端內最大 *bf* 影格與－交換的客戶端內中最大 *bf* 影格必須) >= 交換客戶端的 *bp* 空間

假設 *bf* 代表在一時間單位，Client 正在觀賞的影片和未來會觀賞到的影片內容所佔的影格數; *bp* 代表在一時間單位，Client 已經觀賞影片的影格數。 條件 3 是為了確保進行交換的動作以後，客戶端只需要額外請求短暫時間的上傳頻寬，並使得整個交換後的複雜度降低。如圖 12 所示，*Client B* 和 *Client C* 沒有達到被交換的客戶端內最大的 *bf* 影格與 - 交換的客戶端內中最大 *bf* 影格必須 >= 交換客戶端的 *bp* 空間的條件，如果進行交換，將使 *Client A* 一直上傳二個

頻寬，在 *time* = 6 中 ， *Client C* 拿到影格 4 的內容，而 *bp* 空間不夠提供和 *Client B* 的影格差距，將導致 *Client A* 必需一直提供上傳頻寬給 *Client C*。

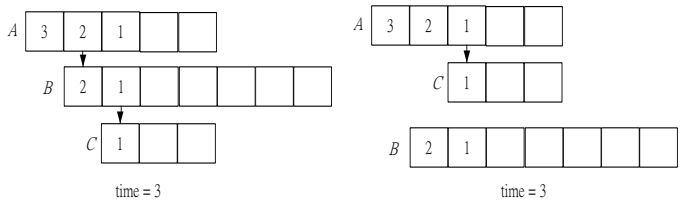


圖 10 ExtLoopback-MDC 條件考慮一

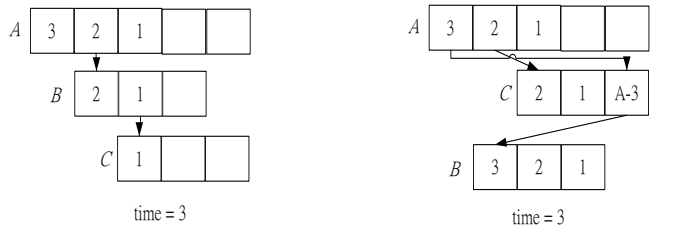


圖 11 ExtLoopback-MDC 條件考慮二

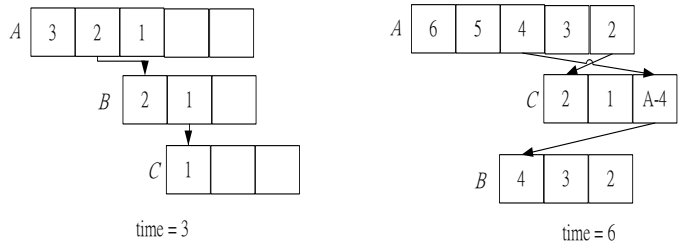


圖 12 ExtLoopback-MDC 條件考慮三

四、ExtLoopback-MDC 問題探討

當使用 Extloopback-MDC 機制的時候，雖然可以使迴圈的可用度提高，服務更多的客戶端，但如果客戶端中途離開，可能產生更多的迴圈斷裂。為了方便觀察此現象，在這節我們將討論在 Loopback-MDC 使用 Extloopback-MDC 機制和沒有使用 Extloopback-MDC 機制下，客戶端的中途離開是否會使伺服器 負載變大。

Loopback 使用 Extloopback-MDC 機制時，當客戶端的中途離開，會使客戶端產生二種狀態。

- **未調整過的客戶**

如圖 13 中的 *Client A*、*Client D* 和 *Client E*，表示沒經過 Extloopback-MDC 機制後的客戶端。從圖 12 可以發現未調整過的客戶端，跟原本的 Loopback-MDC 機制下的客戶端群相同，所以這些客戶端在中途離開所造成的客戶端跟未使用 Extloopback-MDC 機制的情況一樣，而 Sewing 與 Patching 修復機制也可以順利降低伺服器 的負載

- **調整後的客戶端**

如圖 13 中的 *Client B* 和 *Client C* 表示經過 Extloopback-MDC 機制後的客戶端;從圖 13 中可以發現調整過的 peer 群，跟原本的 Loopback-MDC 機制下的客戶端群不相同，所以這些客戶端在中途離開所造成的客戶端跟未使用 Extloopback-MDC 機制的情況不一樣，這些客戶端將可能影響伺服器 的負載。而在經過 Sewing 和 Patching 的調整之後，對於這些客戶端對伺服器端的影響又可以分成兩種客戶端：

1. 調整後的客戶端之儲存空間短的客戶端

如圖 12 中的 *Client C* 它的儲存空間只有 9 個時間單位，在滿足之前描述的三個條件後進行調整，*Client C* 完全可以由 *Client A* 提供影格，又滿足當初第三條件，即表示 *Client A* 只需在短暫時間內，額外提供上傳頻寬給 *Client C* 即可。所以就 *Client A* 和 *Client C* 來看，它一樣遵循著 Loopback-MDC 原有的傳遞影格機制，所以我們可以把這類客戶端發生中途離開的情況，一樣劃分在錯誤修復機制內可能順利修復的客戶端。

2. 調整後的客戶端之儲存空間長的客戶端

如圖 13 中的 *Client B* 它的儲存空間有 9 個時間單位，遠大於 *Client C* 的儲存空間，所以當 *Client B* 離開而造成 loop 斷裂時，將因客戶端之間的下載時間點差太多，而造成錯誤修復將無法

有效降低伺服器的負載。

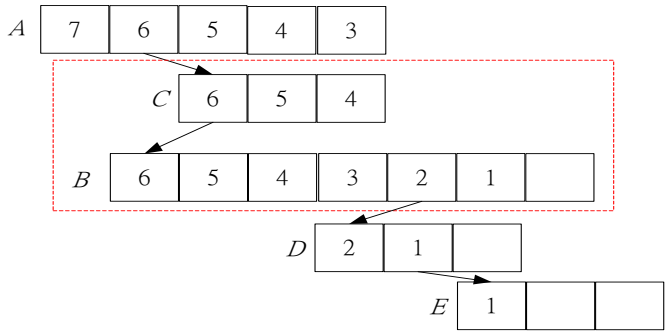


圖 13 ExtLoopback-MDC 機制

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				
d3		B-3	C-2		D-2	E-1	
d4		B-4			D-3	E-2	

(a)

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

(b)

圖 14ExtLoopback-MDC 進行錯誤修復

如圖 **14** 所示，假設每個客戶端儲存空間為 n 個時間單位，因此當有 n 個時間單位沒有客戶端銜接時，將造成迴圈斷裂。每個客戶端依其所需品質來決定它持有的 description 數，如圖 **14**，*Client B* 請求最佳品質，即持有四條 description(*B-1*,*B-2*,*B-3*,*B-4*)。圖 **14** (a)說明當 *Client D* 中途離開系統時,description 1 和 description 2 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 **14** (b)所示，在利用 Sewing 修復時，我們發現因為 *Client C* 所持有的影格內容，已經沒有 *Client E* 所需要的內容，所以我們沒辦法利用 Sewing 修復斷裂的迴圈，接下來使用 Patching，我們發現無法將客戶端 *E-1*,*E-2* 搬移到別的迴圈中，因為 *Client B*、*Client C* 沒有辦法提供 *Client E* 所需要的影格內容，所以 Sewing 和 Patching 無法進行有效修復機制，伺服器便需要額外提供兩條上傳頻寬給予 *Client E*。這是因為一開始為了延長等待新進客戶端的加入時間，所以我們將擁有較長儲存空間客戶端往下交換，所以一但此客戶端中途離開，迴圈前後客戶端所需的影格內容差距變大，而造成錯誤修復不能有效率的降低伺服器負載。

五、效能評估與分析

本章節中將針對 description 分配的模擬實驗結果進行分析及評估。本模擬實驗中，模擬時間為 2000 個單位時間，並使用 MDC 將原始影片的串流編碼為 16 條 description。由於 Loopback-MDC 是建置於同儕式網路上的隨選視訊服務，其透過客戶端將影片片頭暫存，以服務新進客戶端。若客戶端對影片觀看品質的需求為最佳品質，即為 16 條 description，故客戶端所持有的 description 對客戶端的服務能力會產生直接的影響。由於 ExtLoopback-MDC 是建置於同儕式網路上的隨選視訊服務，透過客戶端將影片片頭暫存，以服務新進客戶端，故節點的影片片頭暫存在開放式迴圈對客戶端的服務能力會產生直接的影響，亦即，評估是否片頭越長，越能有效率地降低伺服器端額外付出的頻寬資源。

5.1 不同到達率之伺服器頻寬耗用

圖15為Loopback-MDC和ExtLoopback-MDC在不同的低到達率下，伺服器的消耗頻寬。從圖 14 中可以發現在到達率為 1 時，2 種架構是相同的，那是因為儲存空間的大小在到達率高的情況下，每個客戶端都能順利接上，所以影響不大。而在到達率開始偏低的情況下，如圖 15 中的 1/2，透過 ExtLoopback-MDC 機制，開始有效的降低了一些上傳頻寬，隨著到達率越來越低，如圖 15 中的 1/10，ExtLoopback-MDC 機制比

Loopback-MDC 機制少了 50%的上傳頻寬。在圖 15 中可得知，當到達率越高時，所需伺服器的上傳頻寬越小。因為 description 迴圈中的客戶端，在短時間內即可有下一個客戶端加入，因此所有 description 的迴圈皆維持開啟的狀態，可更加節省伺服器的上傳頻寬。當使用者儲存空間越大時，伺服器需耗用的上傳頻寬也越小，因為 description 迴圈中的客戶端有較長的時間來等待下一個加入的客戶端。從圖 15 中我們可以發現，在到達率越低的情況下，ExtLoopback-MDC 就可以更有效的降低伺服器的上傳頻寬，這表示片頭停留在迴圈的時間比在 Loopback-MDC 機制下更久，有效率降低伺服器負載，提高伺服端的可用度。

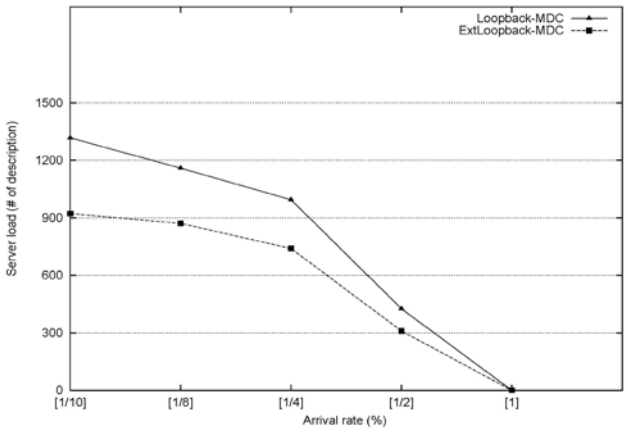


圖 15 不同到達率之伺服器上傳頻寬

5.2 不同丟棄率之伺服器頻寬耗用

圖 16 為 Loopback-MDC 和 ExtLoopback-MDC，在不同的客戶離開率下，伺服器的耗用頻寬。此兩種演算法分別皆採用 sewing 及 patching 做錯誤回復，標示為 Loopback-MDC-sp 和 ExtLoopback-MDC-sp。如圖 **16** 所示，在丟棄率分別為 10%及 20%時，ExtLoopback-MDC-sp 比 Loopback-MDC-sp 分別多消耗 9%及 10%的上傳頻寬，而在丟棄率 30%下，ExtLoopback-MDC-sp 則比 Loopback-MDC-sp 多上傳了 5%的上傳頻寬。我們可以發現，在丟棄率 10%-30%之間，為了延長片頭停留在迴圈中的時間，ExtLoopback-MDC -sp 每次都把影格較長的客戶端往後挪，導致客戶端之間重疊的影格比例變小，而降低了採用 sewing 及 patching 修補成功的機率，因此會比 Loopback-MDC-sp 耗用較多的頻寬。而在 50%的丟棄率時，ExtLoopback-MDC–sp 及 Loopback-MDC –sp 有相同的上傳頻寬，表示在到達率低、丟棄率高的情況下，ExtLoopback-MDC-sp 和 Loopback-MDC-sp 因為服務客戶端都降低，使用者所需的 description 都由伺服器提供，因此兩種架構所消耗的上傳頻寬相同。所以從圖 16 中我們可以得知，ExtLoopback-MDC-sp 在有使用者中途離開時，其伺服器所需負擔的上傳頻寬較 Loopback-MDC-sp 的伺服器多。

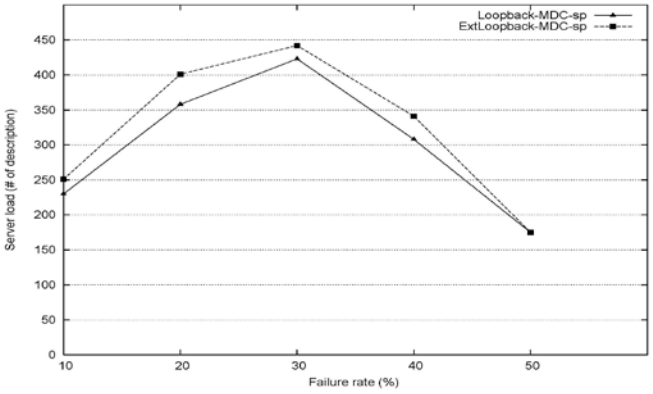


圖 16 不同丟棄率之伺服器上傳頻寬

六、結論

本研究我們提出了新的 ExtLoopback-MDC 機制，利用每個使用者的儲存空間大小不一致和進入的時間點不同，透過交換機制在不影響其他使用者的觀看品質下進行交換。因此片頭可以更有效率的保留在迴圈中，使新進使用者可以順利銜接且降低伺服器上傳頻寬的耗用，進而提高伺服器的可用度。最後經由模擬實驗結果得知，提高伺服器的可用度與增加其強韌度，在本架構中

是不可兼得的(tradeoff)。但是 ExtLoopback-MDC 機制在到達率越低時，可以更有效的降低伺服器的上傳頻寬，使伺服器可以服務更多的使用者，增加其可用度，不過所要付出的代價就是降低其強韌度。在未來研究中，我們將進一步探討在 ExtLoopback-MDC 上如何執行錯誤修復機制，並提出更有效率的修復法，來降低伺服器額外提供上傳頻寬的可能性。

七、參考文獻

[1] 林朝興, 李明憲, 王鼎超, "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, " 2009 資訊科技國際研討會(AIT), 2009

[2] 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,pp.700-711, 2007。

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

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

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