# 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性

## Loopback-MDC: Exploiting Collaborative Caches with Multiple Description Coding to Enhance the Streaming Scalability on CDN-P2P Network

林朝興
南台科技大學 資訊管理研究所
mikelin@mail.stut.edu.tw

李宜婷
南台科技大學 資訊管理研究所
m9590208@webmail.stut.edu.tw

## 摘要

由於網際網路的蓬勃發展，資訊可快速地被傳遞，只有文字和圖片的資訊交流，已無法滿足使用者的需求。隨著電腦相關科技的逐漸進步，如何有效地提供多媒體串流服務內容給各種不同下載頻寬的廣大群眾已經成為一項值得注意的議題。本研究主要討論在 CDN-P2P(Content Distribution Network－Peer to Peer)混合架構模式下，以 loopback 機制運用 MDC(Multiple description Coding)多重描述編碼技術的概念，來解決節點網路頻寬異質性的議題，讓使用者可依需求來享有不同的隨選視訊觀看品質。本論文中，我們提出 OLF(Open-Loop-First) 方法，在分配串流描述(description)時，考量 Loopback 機制的堅韌度(robustness)和可用度(availability)，來提升代理伺服器(proxy server)的延展性，以提供使用者更好的多媒體傳輸服務。模擬實驗結果顯示 OLF 串流描述分配方法在 loopback-MDC 的機制上可以有效地降低 CDN-P2P 隨選視訊系統代理伺服器的上傳頻寬。

***關鍵詞：****回傳迴圈, 多重描述編碼, 同儕式網路, 代理伺服器, 開放迴圈優先*

## Abstract

Owing to the widely prevailing deployment of the Internet, the information now can be rapidly disseminated all over the world. Only the content of static texts and pictures has no longer met users' demands. Along with the speedy advance of computer-related technologies, how to efficiently provide multimedia content for the large various crowd on the Internet, especially videos, becomes the noticeable issue. In this paper, we discuss the adaptation of MDC (Multiple description Coding) for the loopback buffering mechanism to address the issue of peer heterogeneity on CDN-P2P (Content Distribution Network–Peer to Peer) video-on-demand system. The proposed OLF (Open-Loop-First) description selection policy for a new peer takes the robustness and the availability of loopback-MDC mechanism into account to raise the scalability of proxy server. The simulation results show that the OLF with loopback-MDC can effectively reduce the amount of uploading bandwidth of a proxy on CDN-P2P video-on-demand system.

*Keywords: loopback, multiple description coding, peer-to-peer network, proxy server, open-loop-first*

## 一、前言

VoD server 主要希望提高系統的延展性(scalability)，當有大量的觀看請求到達(flash crowd)時，server 依然可以進行服務，這考驗著 server 的頻寬和儲存空間是否足以解決突來的大量請求。過去的研究中，在架設 VoD server 考量各種的基本架

構來提升 VoD server 的效能。

傳統的 Client-server 架構[5][8][14]，主要由 server 各分配專屬的 channel 來滿足所有 clients 的要求，屬集中式控管。沒有延遲服務的問題，且所接收到的串流品質較可靠。但當有大量請求發生，系統無法負荷時，則導致 server 當機或無法正常運作。後來許多學者提出其他改善的做法，像是 Batching[1]、Patching[9]等技術，但上述的技術皆建立在 IP multicast 上，牽涉到路由器的設定，依目前網路架構而言，難以實做。因此後來研究提出 CDNs (Content distribution networks)架構的概念，CDNs 是在網路基礎建設中，各區域網路上設立多台 proxy servers，並將影片內容放置於 proxy servers 中，當有請求到達時，由 proxy servers 來共同分擔及滿足 client 的請求，避免造成單一 server 的負載過重。但是在 CDNs 架構上要增加延展性，需大量設置 proxy servers，因此需投入相當可觀的資源和成本。後來的 Peer-to-peer 架構[3][12]可解決大量請求湧入，避免系統無法負荷。運用 peer 間合作的概念，peer 是接收端也是服務端，利用 client 的儲存空間來暫存影片片段，其頻寬為其他的 client 傳送暫存的影片片段，讓 client 的儲存空間和頻寬，支援 server 的有限資源，降低 server 負載。因此大量請求到達，請求的 client 數量增加，可支援 server 的資源同時也增加，如此便可增加 VoD server 的延展性。

Peer-to-peer 架構雖然滿足 server 延展性的需求，但存在服務不穩定的缺點。因為 client 的加入和離開，並不是 server 所能預期，當提供服務的 client 中途離開系統，需採取錯誤回復(failure recovery)機制，讓其他 client 不受 client 背棄而中途無法觀看影片。依以上的原因，有學者提出了 Loopback 機制[6]應用在 CDN-P2P 混合架構上，以結合現有架構的優點，並屏除缺點。且 CDN-P2P 在[19][20]的研究成果上，已經證實可以有效地提供高擴充性的串流服務。

CDN-P2P 混合架構 [17] 中有 central server (中央伺服器)、proxy servers (代理伺服器)，如圖 1。central server 儲存所有影片的內容，提供 proxy servers 影片內容及當 proxy servers 資源不足時，替代 proxy servers 為 client 提供服務。proxy servers 設立於各區域網路上，負責 cache 熱門影片的前半段。當 proxy servers 負責區域的 client 發出請求，proxy server 若有暫存 client 請求的影片，則 proxy server 來傳送 clients 所需影片串流。proxy server 暫存熱門影片，因此可為大部分的請求直接提供服務，減輕 central server 的負載，且同區域網路的串流傳送，有更穩定和更快速的傳送品質。CDN-P2P 混合架構中，同區域的 clients 組成 local peer network 相互支援，善用 clients 的資源，讓 client 為另一個 client 進行服務，進而擴展 server 整體的延展性。且有 server 進行控管，避免產生延遲服務的現象，且當傳送過程中不順利時，也方便進行回復錯誤的程序。

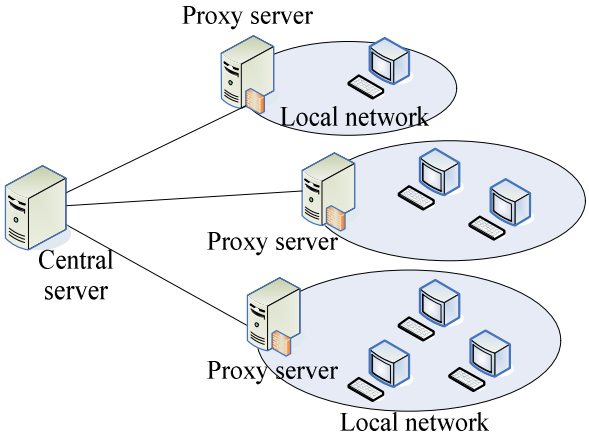


圖 1 CDN-P2P 網路架構

Loopback 機制針對 VoD 的需求特性進行設計，利用 client 間合作關係進行串流分享，且只要讓影片片頭保留在 client 的 buffer 中，即可讓 client 為下一個新加入的 client 進行服務。當請求增加，到達率密集，則影片片頭保留在 client 的 buffer 中的機率越大，便可利用 client 的資源來服務更多的請求。

從上面描述中，可發現 Loopback 機制

在 VoD 服務下，有快速平穩傳送串流和增加 server 延展性等優勢。且 Lookback 機制串連請求相同影片的 client，如同 chaining 的架構[18]，依序進行串流傳遞。有別於一般樹狀結構，一個 peer 需提供多個 peer 串流資料。所以整體的串流結構，鏈狀比樹狀結構來得簡單。但 Lookback 機制只提供單一版本的串流資料，使得 peer 間實際運作多媒體傳輸受到限制。因為 peer 間存在著網路異質性，只提供單一版本的串流，造成符合頻寬條件的 peer 才能夠順利地進行多媒體的傳輸。或者 server 把影片設為最低的串流品質，才能讓多數的 peer 得以順利傳送串流資料，如此不適性化的缺陷，使得 Lookback 機制，其應用的價值驟減。本研究為上述的問題尋找一個解決之道，提出了在 Lookback 機制下，施行 MDC 編碼技術，讓 client 可以依照頻寬限制或選擇想要的觀看品質，請求不同的串流品質，並且考量較佳分配 description 的方法，更有彈性及可靠的進行串流傳送。

論文章節安排如下，在第二節中，探討相關研究的文獻。第三節為編碼後，如何適當的進行 description 分配。第四節為研究的實驗結果。第五節為最後的結論與未來發展。

## 二、文獻探討

### (一) Lookback

Lookback 採用 CDN-P2P 混合架構，假設 client 有暫存的能力，且動態暫存影片片段於 buffer 中。且 client 可以在同時段下，對外傳送一條 stream。client 的合作時間只有在撥放此部影片和播放後一段時間。client 在進行 stream 傳送時，有隨時發生傳送失敗或選擇中途離開 server 的可能。

Lookback 機制有 central server 和 proxy servers，每個 proxy server 設置在區域網路上，並且 proxy servers 需暫存熱門影片的前半段部分，當有 client 送出請求時，由 proxy server 提供串流服務，如果 proxy server 無法滿足 client 的請求，或 client 已觀看到影片片尾時，則由 central server 來提供影片內容。在同一區域網路中，要求觀看相同影片的 client 會被 proxy server 依照請求的時間組成一個 Loop。

proxy server 將影片內容移轉給 Loop 中第一個到達的 client，由 client 的 buffer 暫代 proxy server 儲存影片片段，proxy server 將可把空出來的儲存空間，儲存影片後半段的片段。當下一個 client 到達並請求相同的影片，若第一個 client 的 buffer 尚存有影片片頭，則把影片串流傳送給 Loop 裡的下一個 client。在 Loop 中的最後一個 client 的 buffer 已滿，但沒有出現相同請求的 client，最後一個 client 就會把觀看完的影片片段傳回 proxy server。當 client 在觀看影片後半段部分時，需由 central server 來提供影片片段，且在 proxy buffer 空間足夠的情況下，Loop 最後一個 client 將影片片段轉送到 proxy server，使後來形成的 Loop 在觀看片尾部分時，不必再向 central server 請求資料，直接從 proxy server 取得，此機制在大量請求湧入時，有效減少 central server 的負載量。如圖 2 中的 loop a 從 central server 請求資料後，傳回 proxy server 後，可讓 loop b 在觀看後半段影片時，直接向 proxy server 請求。

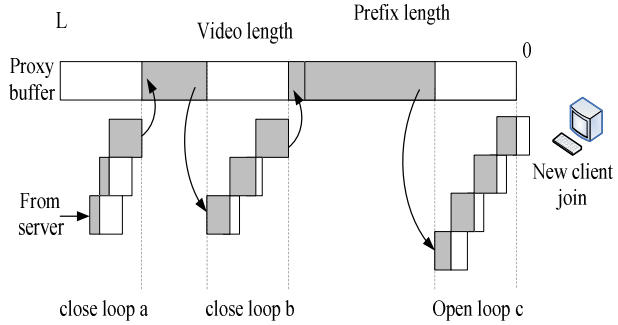

圖 2 Loopback 運作示意圖

在 Lookback 機制中，Loop 運作中有二種情況，如下：

1. Open Loop

Loop 呈現可讓新的 client 加入的開啟狀態，如圖 2 的 Loop c。Loop 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。

2. Close Loop

Loop 呈現無法讓新的 client 加入的關閉狀態，如圖 2 的 Loop a。Loop 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 加入則需開啟另一個新的 Loop。

因此在 Lookback 機制中，讓 Loop 保持開啟的狀態，即可有效降低 server 所需提供的上傳頻寬，減輕 server 負載，進而提升系統的延展性。

Lookback 中大部分由 client 相互提供服務，因此需要考慮當 client 中途離開時，如何修復 client 離開所造成子孫節點的 client 無法繼續正常接收串流的問題。Lookback 的錯誤回覆(Failure recovery)中，利用 client buffer 中暫存的資料進行修復，以解決上述的問題。如在某 Loop 中，其 peer 傳送順序為 peer A、peer B、peer C，peer A 擁有影格 10 至影格 7，peer B 擁有影格 8 至影格 6，peer C 擁有影格 6 至影格 4，當 peer B 離開 server 時，peer A 可以為 peer C 提供所需的影格 7 以後的串流，因此 Loop 的運作並不會受到 peer B 離開，而影響其他 peer 的影片觀看。則可提升 loop 的堅韌度，增加 client 觀看影片的穩定性。關於錯誤回覆的細節部分並不在本篇討論的範圍內。

(二) MDC 編碼技術

近期，多媒體串流探討網路異質性議題。網路異質性，指網路上各個 client 所持有的資源與頻寬不一的情況，造成每個 client 所要求的觀看品質有所不同。因此如果 server 只提供單一版本的串流，並無法滿足 client 端多樣化的請求，且 client 間因頻寬不同，難以分享與傳送串流。

為解決上述議題，已有相關編碼技術被發表。編碼技術主要是將一條完整影片串流，編碼成多條串流，當 client 所接收到的串流數量越多，則代表 client 獲得的影片觀看品質越好。因而可視 client 的頻寬來調整欲下載的串流數量，即可解決上述的問題，目前常見的編碼技術分為 Layer encoding[4][11][16] 和 MDC encoding[2][7][10][13][15]。

MDC 編碼技術是將完整的影片編碼成多條的串流，每一條串流為『description』。每條 description 並不具有相依性及順序性的限制，只要任意接收不同編號的 description，所獲得的封包內容越多，即可提升 client 的觀看品質，因此可以更有彈性的調整影片品質。

## 三、Description 分配機制

在 loopback 機制下進行 MDC 編碼，將影片串流編碼為多個 descriptions，並且加以編號，client 依頻寬限制，請求適當的 description 數量。

圖 3 為 Loopback-MDC 運作示意圖。假設一部影片分成了 3 條 descriptions，則會產生 3 條 loop，client 依請求的數量，加入 description 的 loop 當中。如圖 3，當 client 請求接收最佳的觀看品質（full quality），則需加入 descriptions 1、descriptions 2 和 descriptions 3 中的 loop 1，取得完整的觀看品質。

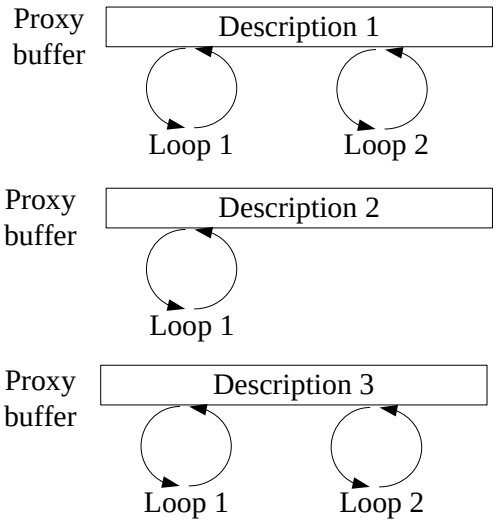


圖 3 Loopback-MDC 運作示意圖

Loopback 機制加入 MDC 編碼技術的應用，需考慮在 client 請求不同的 description 數量下，如何有效的分配 description 給 client，讓 description 中的 Loop 能夠保持開啟的狀態，使 client 間有效的進行串流分享的傳送，減少 server 的負載。且在分配 description 時，如何使 loop 更加堅韌，讓失敗的 peer 中途離開時，不會造成系統上突然過高的負載，讓 loopback 機制能夠發揮最大的效益。

- ORD (Ordered description 分配)

ORD 依照新加入的 peer 請求 description 的數量，將 description 的編號由小至大，進行分配 description。

圖 4 為 ORD 範例，假設一部影片共分四條 description，$d_1$、$d_2$、$d_3$、$d_4$，有五個 peer 請求相同的影片，為 A、B…E，其請求到達的次序，假設一個方格代表一個時間單位。當 peer A 請求二條 description 的觀看品質，以 description 編號由小至大進行分配，於是先分配 description d1 給 peer A，再分配 description d2，圖 4 中 A-1 表示 peer A 分配的第一條 description，其數字表示分配 description 的優先次序。

當 peer 的頻寬需求相同時，可節省上傳頻寬數量，但 peer 的頻寬需求差異大時，description 編號較大的 loop，因為少有 peer 加入，會形成多個擁有單一或少數 peer 的 loop。因此，ORD 分配較適用於 peer 的頻寬需求大致相同的環境。

Time →

| | | | | | |
|---|---|---|---|---|---|
| $d_1$ | A-1 | B-1 | C-1 | D-1 | E-1 |
| $d_2$ | A-2 | B-2 | C-2 | D-2 | E-2 |
| $d_3$ | | B-3 | C-3 | | E-3 |
| $d_4$ | | | | | |

圖 4　ORD description 分配範例

- RR (Round-Robin description 分配)

RR 分配是讓每條 description 都平均分配給 peer，讓每條 description 平均被 server 中的 peer 所持有。當有 client 請求發生，則依 description 編號依序輪流進行分配。

圖 5 為 RR 分配範例，當 peer A 請求二條 description 的觀看品質，分配 $d_1$ 和 $d_2$ 給 peer A。peer B 到達並請求三條 description，server 接續分配 $d_3$ 和 $d_4$ 給 peer B，再分配 $d_1$。當 peer C 請求到達時，再接續分配 $d_2$，以此類推。

在 RR 分配中，當請求的到達率較高時，所有 description 中的 loop 仍維持開啟的狀態，可減少 server 上傳頻寬數量，且當有 peer 請求最佳觀看品質(full quality)時，也不需由 server 來提供資源，peer 間即可滿足其請求。但是當 peer 的到達率低或平均 peer 需求品質低時，server 提供的上傳頻寬數量反而增加，因為低到達率讓 peer 間 buffer 內容無法銜接，loop 由開啟狀態轉為關閉狀態，使下一個 peer 在加入後，需開啟新的 loop，如此 server 提供的上傳頻寬數量反而增加，其負載也增加。所以 RR 分配較適用於 peer 高到達率，且請求觀看品質不一致的情況。

Time →

| | | | | | |
|---|---|---|---|---|---|
| $d_1$ | A-1 | B-3 | C-4 | | E-2 |
| $d_2$ | A-2 | | C-1 | D-1 | E-3 |
| $d_3$ | | B-1 | C-2 | D-2 | |
| $d_4$ | | B-2 | C-3 | | E-1 |

圖 5　RR description 分配範例

- OLF(Open-Loop-First description 分配)

OLF description 分配，主要在進行 description 分配時，需以 open loop 為優先分配的對象。如一部影片共有 5 條的 descriptions，2 條為 open loop，3 條為 close loop，一個新加入的 peer，共要求 3 條 descriptions 的觀看品質，因此系統會先優先分配 2 條 open loop 的 description 給新加入的 peer，在 loop 為關閉狀態的 description，開啟一條新 loop，讓新加入的 peer 加入，以達到欲觀看的品質。

OLF 先找出為 open loop 的 description，再以這些 description 進行分配，讓這些 loop 維持開啟的狀態，因為新開啟一個 loop，需花費一條 proxy server 的上傳頻寬。如公式(1)，找出 open loop 的 description 集合，假設 $O$ 為 open loop 的 description 的集合，當 loop 最後一個 peer 的 buffer 結束時間大於或等於 new peer 加入時間(*NowTime*)，則為 Open loop 的 description。且影片共分為 $m$ 條的 description，$n_d$ 為 description $d$ loop 中所有的 peer 數量，$j_d$ 為 description $d$ 的 loop 中最後一個 Peer，$T_{d,n_d}$ 為 description $d$ 的 loop 中最後一個 Peer 的加入時間，$B_{d,n_d}$ 為 description $d$ 的 loop 中最後一個 peer 的 buffer 結束時間。

$$O \leftarrow \{\, d \mid NowTime \le (T_{d,n_d} + B_{d,n_d}),\ 1 \le d \le m \,\} \quad (1)$$

圖 6 為 OLF 分配範例，分配 $d_1$ 和 $d_2$ 給 peer A 後。peer B 到達並請求三條 description，server 優先分配 open loop 的 description $d_1$ 和 $d_2$ 給 peer B，無法滿足 peer B 請求的三條 description 的請求，所以再開啟 $d_3$ 的 loop 分配 $d_3$ 給 peer B。如在第二節所提到的，在 loopback 機制中，盡量保持 loop 為開啟的狀態，以利新的 peer 加入原有的 loop 中，可顯著地節省 server 頻寬的問題，因此當有請求發生時，優先分配 opne loop 的 description，讓 loop 盡量維持開啟的狀態。

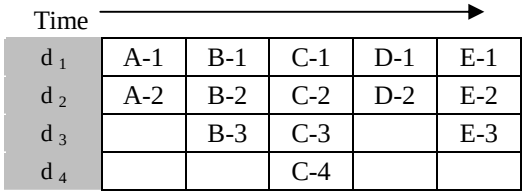


圖 6 OLF description 分配範例

- OLF－分配 descriptions 的優先權

為了使 OLF descriptions 分配加長 description 的 loop 維持開啟的時間和加強 loop 的堅韌度，使其他 client 不因中途離開的 client 而造成觀看中斷。本小節加入了分配 descriptions 的優先權的討論，當無法決定 description 分配時，先滿足延長 open loop 策略後，再考慮加強 loop 堅韌性，進行 description 的分配。

1. 延長 open loop 策略

在 description 的 loop 中最後一個 peer 尚有剩餘的 buffer 空間，且為所有 description 的 loop 最後一個 peer 剩餘 buffer 空間最小者，為優先分配的 description。公式(2)，集合 *ME* 為進行比較的 description 的 loop 中，peer 剩餘 buffer 空間最小的 description 集合，*dec* 為進行比較 peer 的範圍，當 peer 的加入時間加上 buffer 長度，則為 peer buffer 結束時間。其考量為把快成為關閉狀態的 open loop，加入新 peer，利用 peer 的 buffer 再延長其 loop 開啟的狀態，維持 loop 的可用性，避免成為 close loop。

$$ME \leftarrow \{\, d \mid d \in \arg\min_{d \in E}(T_{d,n_d\text{-}dec} + B_{d,n_d\text{-}dec})\} \quad (2)$$

圖 7 延長 open loop 策略範例，假設 description $d_1$ 和 $d_2$ 的 loop 皆為開啟的狀態，已有 peer A 和 peer B 的請求到達，所有 peer 的 buffer 大小皆為二個時間單位，$d_1$ 的 loop 中最後一個 peer buffer 結束時間為 $T_2$，$d_2$ 為 $T_1$，則當 peer C 請求 description 時，優先分配 $d_2$，因為 $T_2 > T_1$，$d_1$ 的 loop 中最後一個 peer buffer 剩餘時間較少，因此優先分配 $d_1$，以免 $d_1$ 的 loop 成為關閉的狀態，無法再接續下一個 peer 的請求。

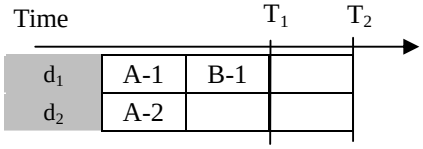


圖 7 延長 open loop 策略範例

2. 加強 loop 堅韌度策略

在考量延長 open loop 策略後，若未能決定分配的 description，則再考慮加強 loop 堅韌度策略，進行 description 篩選。加強

loop 堅韌度策略主要讓 description 的 loop 中 peer 平均加入各個 description 的 loop 中，讓 description 的 loop 中 peer 間 buffer 重疊資料量一致。如某些 loop 中的 peer 較密集，則較密集的 loop，因為 peer 間 buffer 的資料較容易重複，因此當有 peer 中途離開時，loop 可利用尚存的 peer buffer 中重複的資料，傳給所需要的 peer，使 loop 較不易斷裂，加強 loop 的堅韌度。反之，當某些 loop 的 peer 較分散，其 loop 的堅韌度較差。因此讓 peer 平均加入各個 description 的 loop 中，可提升整體的 loop 的堅韌度。加強 loop 堅韌度策略，比較 loop 中倒數第二個 peer 的 buffer 結束時間較早的 description，當符合條件的 description 數量依然多於 peer 欲請求的數量，再比較 loop 中倒數第三個 peer，以此類推。

圖 8 為加強 loop 堅韌度策略範例，$d_1$ 和 $d_2$ 的 loop 中最後一個 peer C buffer 結束時間皆為 $T_3$，所以比較前一個 peer buffer 結束時間，$d_1$ 倒數第二個 peer 為 B，buffer 結束時間是 $T_2$，$d_2$ 倒數第二個 peer 為 A，buffer 結束時間是 $T_1$，因為 $T_2 > T_1$，因此優先分配 $d_2$。

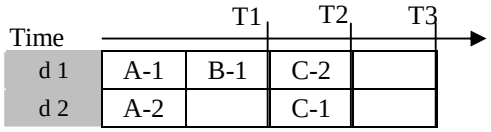


圖 8 加強 loop 堅韌度策略範例

加強 loop 堅韌度策略最主要是為了加強 loop 的堅韌度，使得 peer 中途離開，其他 peer 較不容易造成觀看中斷。以圖 9 進行說明，假設每個 peer buffer 長度皆為二個時間單位，當有連續三個時間單位沒有 peer 加入時，將使得前後 peer 無法銜接，而導致 loop 斷裂。圖 9 (a)為只考慮最後一個 peer buffer 的排列情況，圖 9 (b)為除了考慮最後一個 peer buffer 外，還考慮之前祖先 peer buffer 的排列情況，圖中呈灰色的 peer，指這些 peer 離開 server 時，將會造成 loop 斷裂，圖 9 (a)中共四個 peer，C-1、D-1、E-2、F-1，圖 9 (b)只有一個 D-1，因此當有 peer 中途離開 server 時，圖 9 (a)的 peer 排列會比圖 9 (b) 的 peer 排列更容易發生 loop 斷裂的情況，使得 server 需花更多的資源進行錯誤回復，為受影響的 peer 提供上傳頻寬。

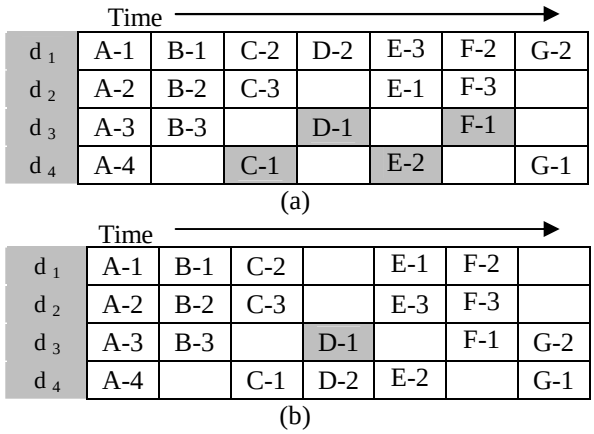


圖 9 考慮祖先 peer buffer 排列的比較

在進行加強 loop 堅韌度策略，比較祖先 peer buffer 需符合其 buffer 結束時間大於新加入 peer 的加入時間。因為當祖先 peer 的 buffer 結束時間早於新 peer 的加入時間時，新 peer 的加入並不會為此祖先 peer 帶來影響。因為雙方沒有任何資料重疊的可能，此祖先 peer 中途離開並不會直接影響新加入 peer 的影片觀看，因此沒有考量的必要。公式(3)中，集合 $E$ 為為搜尋所有 loop 中，倒數第 $des$ 個祖先 peer 可銜接新加入 peer 的 description 的集合， 即需 $(T_{d,n_d\text{-}dec} + B_{d,n_d\text{-}dec}) \geq NowTime$ 。 $Bd,i$ 為 description $d$ 的第 $i$ 個 Peer 的 buffer 長度; $Td,i$ 為 description $d$ 的第 $i$ 個 Peer 的 加入時間。

$$E \leftarrow \{d \mid d \in \arg(T_{d,n_d\text{-}dec} + B_{d,n_d\text{-}dec}) \geq NowTime\} \quad (3)$$

圖 10 為需考量 buffer 的祖先 peer 範例。假設 $T_{now}$ 為現在的時間，也是下一個 peer E 的加入時間，每個 peer 的 buffer 長度都是二個時間單位。peer D buffer 結束時間為 $T_d$，$T_{now} < T_d$，所以 peer D 需列為祖先 peer 的 buffer 比較名單，peer C buffer 結束時間為 $T_{now}$，因此 peer C 也需進行比

較，peer B buffer 結束時間為 $T_b$， $T_b < T_{now}$，peer B 和 peer E 無法銜接，因此在 peer B 以前加入的 peer(包括 peer B)不被進行比較。圖 10 程序如下，當 peer E 要求一條 description，進行 description $d_1$、$d_2$ 比較，選擇其一加入，$d_1$ 和 $d_2$ 中最後一個 peer D buffer 結束時間相同，因此往前一個祖先 peer C 比較，$d_1$ 和 $d_2$ 中 peer C buffer 結束時間相同，再往前一個 peer 比較，$d_1$ 的 peer B buffer 結束時間 $d_2$ 的 peer A buffer 結束時間皆小於 peer E 加入時間，因此不需再比較，而 peer E 只要任選 $d_1$ 和 $d_2$ 中某條 description 即可。

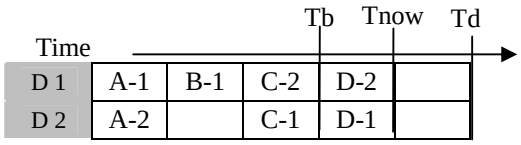


圖 10 需考量 buffer 的祖先 peer 範例

- OLF 演算法

OLF 分為 3 步驟，分述如下。

1. 找出 Open loop

在 OLF 中，進行分配 description，需以 open loop 的 description 為優先分配的對象。因此 peer 加入系統時，會告知系統 peer buffer 的空間及所要求的 description 數量。$B_{d,i}$ 為 description $d$ 的第 $i$ 個 Peer 的 buffer time，$T_{d,i}$ 為 description $d$ 的第 $i$ 個 Peer 的 join time，$d$ 為一部影片切分 description 的總數量。Server 先找尋目前 open loop 的 description 有哪些，若目前沒有 open loop，則依新加入的 peer 的需求，開啟所需求數量的 loop 數量。若有 open loop 的 description，則進行尋找符合條件的 description 進行分配，集合 $E$。

圖 11 找出 Open loop 演算法，新加入 peer 請求 description 時，系統優先尋找目前 open loop 的 description 的集合 $O$，若 description 數量集合 $O$ 數量等於 peer 請求數量 $ND_{new}$，則直接分配 description 給請求的 peer，且分配的 description 不可為 peer 已持有 $H_{new}$，若集合 $O$ 數量不足再開啟 description 的新 loop 讓 peer 加入，若集合 $O$ 數量大於請求數量，再用加強 loop 堅韌度策略，篩選 loop 中祖先 peer buffer 結束時間較早的 description，進行分配 description，直到滿足使用者的觀看品質的數量。

```
// *  初始化  * //
    m  爲影片的全部的description數量
    集合 E = ∅ ; // 爲open loop,且爲候選loop
    集合O = ∅; //open loop 的description number集合
    n_d = 0 ( d  |  1 ≤ d ≤ m) ; // description 1~m 持有peer的數量
OLF (D_new, B_new){
// *  new peer join    * //
    ND_new ← D_new; // 被滿足的請求description數量
    H_new ← ∅ ; // new peer持有的description數量,起始爲空值
    E ← ∅ ; // 候選的loop之集合
    AE ← ∅ ; // 尚未被選取的候選loop之集合
    dec ← 0 ; //peer遞減指標
    //找出description 1~m 有幾個open loop
    O ← { d | NowTime ≤ (T_{d,n_d} + B_{d,n_d} ) , 1 ≤ d ≤ m };
    if (| O |= 0) // 沒有任何的open loop ,爲第一個first peer
            E ← E ∪{1,2,...,m};
            Allocate ( E );  // 分配description給new peer
     if (| O |> 0) // 有一個以上的open loop
            E ← E ∪ O;
            repeat
                    //尋找上一層的候選loop集合
                    Find_Pre_Loop(dec,  m,  E);
                    //從候選loop中確認的第j - dec個peer條件,
                      是否爲優先分配的description
                    Check_Can_Loop( E ,  ND_new, O, AE);
                    dec ← dec + 1;
            until  ND_new = 0   // 滿足觀看品質
 }
  //*allocation description*//
    Allocate (E) {
    if ( | E | >ND_new)
            ND' = ND_new;
    else
            ND'  =| E | ;
    repeat
          get description d ∈ E;
          ND_new =  ND_new - 1;
          H_new ←  H_new ∪{d  }; // 記錄已持有的description
          n_d ← n_d + 1; // 把description d的peer數量+1
          T_{d,n_d} ← NowTime;
          B_{d,n_d} ← B_new;
      ND'- -;
    until (ND' = 0)
    return ;
   }
```

圖 11 找出 Open loop 演算法

2. 加強 loop 堅韌度策略的候選 description

當集合 $O$ 數量大於請求數量，進行 description 的篩選。則需尋找不同 description 的 loop 中，同順位的 peer，進行加強 loop 堅韌度策略的篩選，進行 peer buffer 結束時間比較，結束時間較早者，其 description 優先進行分配。*dec* 為遞減值是同順位的基準值，*dec* 初始值為 0，指 loop 中最後一個 peer，往前推一位，則加 1。$n_d$ 為 description $d$ 的所有 peer 的數量。當考量延長 open loop 策略，進行分配 description 後，仍然有多條候選 description 的 loop 可為 peer 提供串流，因此再從這些候選 loop 中，再找出祖先 peer buffer 結束時間較短的 description。

圖 12 為加強 loop 堅韌度的候選 description 的演算法，找出祖先 peer buffer 結束時間較短的 description 的集合 $E$，並且保留惕除掉的 description 集合 $AE$，當篩選後的候選 description 小於 peer 所請求的 description 數量，再從其保留的 description 集合 $AE$ 繼續進行 description 分配。

```
Find_Pre_Loop(dec, m, E){
    LE ← E;
    for(d = 1, d ≤ m, d++){
        E ← E ∪ {d | d ∈ arg(T_{d,n_d}-dec + B_{d,n_d}-dec) ≥ NowTime, d ∈ E};
    }
    AE ← LE − E;
    return;
}
```

圖 12 加強 loop 堅韌度的候選 description 的演算法

3. 篩選加強 loop 堅韌度策略的 description

在上一個步驟，找出加強 loop 堅韌度策略的候選 description 後，則進行篩選加強 loop 堅韌度策略的 description。若候選 description 數量剛好滿足新加入的 peer 所請求的數量，則直接進行分配。若候選 description 的數量不足以滿足新加入 peer 所請求的數量，則分配候選 description，不足的部分需要開啟請求 peer 未持有 description 的新 loop。若候選 description 的數量多於新 peer 所請求的數量，則找出符合加強 loop 堅韌度策略的候選 description 的 loop 中，將 peer 的 buffer 結束時間最小者設為新的候選 description。其餘的 open loop 的 description 集合 *AE*，則保留以防止當目前的新候選 description 集合 *E*，其 description 數量無法滿足使用者需求時，做為下回分配的對象。但當目前的新候選 loop 集合 *E* 產生，卻沒有其餘的 open loop 集合 *AE*，則代表此候選 description 的 loop 中的 peer buffer 結束時間一致，需再進行對父 peer 進行進一步的篩選，即 *des*+1，重複步驟 2。當候選 description 為 0，則代表搜尋父 peer 時，沒有符合的條件的 description，因此需以原來的候選 description 集合 *LE* 中，再進行挑選，如圖 13。

```
Check_Can_Loop( E , ND_new, O, AE){
    //比較以前的祖先peer,已無節點比較
    if ( | E | = 0 && | AE |= 0)
        E ← LE;
        Allocate ( E);
    //open loop的數量=peer需求的description數量
    else if ( | E | = ND_new)
        Allocate ( E);
  //open loop的數量<peer需求的description數量
    else if ( | E | < ND_new)
        Allocate (E); //把目前的open loop的數量分配完
        if ( | AE | = 0 )
            // 從close loop中開新的loop
            E ← {d | d ∉ O}
            Allocate (E);
        if ( | AE | > 0 )
            E ← AE;
            AE ← ∅;
            Check_Can_Loop( E , ND_new, O, AE);
    //open loop的數量>peer需求的description數量
    else if ( | E | > ND_new)
        ME ← { d | d ∈ arg MIN_{d∈E}(T_{d,n_d}-dec + B_{d,n_d}-dec),
                d ∉ H_new }
        AE ← {d | d ∈ E, d ∉ ME};
        E ← ME;
        if( | AE | != 0 )
            Check_Can_Loop( E , ND_new, O, AE);
    return ;
}
```

圖 13 篩選加強 loop 堅韌度策略的 description 演算法

## 四、效能評估與分析

以下將利用本研究所提的三個 description 分配方法，ORD、RR 和 OLF，進行模擬實驗。實驗中，設定影片分為 8 條 description。模擬時間為 2000 個單位時間。peer 請求的 description 數量，依照常態分配( $\mu = 5$，$\sigma = 3$ )，評估三項分配方法在到達率和 peer buffers 長度的變化下，server 整體的頻寬利用情況。

圖 14 不同的使用者 buffer 之 proxy server 上傳頻寬的比較。假設到達率為 1，變動使用者的 buffer 長度。當使用者的 buffer 越多，則三種方法的 proxy server 上傳數量越少。當使用者的 buffer 為 1 時，OLF 和 ORD 的 proxy server 上傳數量較少。因 OLF 以 loop 開啟的 description，優先分配，可降低 server 重開新的 description loop 的可能性。ORD 為 description 編號較小者優先分配，因此編號較小的 description，其 loop 維持開啟的可能性越高，則可降低 server 負載。RR 是依 description 編號輪流分配，因此使用者的 buffer 長度太短，即可等待下一個 peer 加入的時間太短，則容易使 open loop 的狀態轉為關閉，所以 RR 的 proxy server 需開啟新的 loop。因此 server 的上傳頻寬較 OLF 和 ORD 多。但當使用者 buffer 增加時，使得 RR 中的 description 中的 loop 有更長的時間去等待下一個 peer 的加入，因此在使用者 buffer 1 至 3 時間單位，RR 的 proxy server 上傳數量驟減。

圖 14 中 RR 方法需使用者的 buffer 大於等於 4，才能和 OLF 有相同 proxy server 上傳數量，因此可得知 RR 需依靠足夠的使用者 buffer 長度，才能發揮其優異的效能。且在使用者的 buffer 為 4 時，三種分配方法皆有足夠的時間，去等待下一個 peer 的加入。

ORD 無法在使用者 buffer 增加時，有效降低 proxy server 上傳數量，因為編號較大的 description 較不容易有 peer 加入，因此容易導致當下一個 peer 加入時，已無法和前一個 peer 接續得上，尤其是整體的 peer 請求 description 的數量差異甚大。

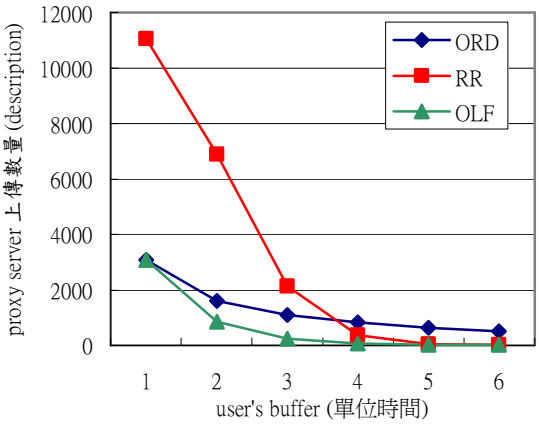


圖 14 不同的使用者 buffer size 之 proxy server 上傳頻寬的比較

在圖 15 中，假設所有使用者的 buffer 長度都是 4 個時間單位，可觀察出 ORD、RR 和 OLF 中，OLF 在不同的到達率下都是最節省 proxy server 上傳數量。

在到達率為 0.125，三種方法所得到的 proxy server 的上傳頻寬皆相同，指的是在極低的到達率下，所有方法中加入的 peer 皆無法接續在前一個請求的 peer，因此 proxy server 的上傳頻寬皆相同。

當到達率為 0.125 至 0.5，ORD 和 OLF 的 proxy server 上傳數量較相近，因為 ORD 由編號較小的 description 依序分配，因此編號較小的 description 容易有 peer 加入。當到達率低時，編號較小的 description 依然可保有 open loop 的狀態。OLF 挑選 description 時，考量 open loop 為優先加入，讓 loop 維持開啟的狀態，可使得 proxy server 減低開啟新 loop 的機會，也是減少 proxy server 的上傳頻寬。而 RR 依 description 編號依序加入 peer，當到達率低時，將導致同一個 description 中的前後 peer 到達時間差異過大，而前一個 peer 的 buffer 已沒有片頭可以提供服務，因此後來的 peer 只能不斷的向 proxy server 發出請求，導致 proxy server 上傳頻寬激增，這種情況當到達率越低越顯著。而 RR 在到達率為 0.125 至 0.5 反而遞減，並不是 proxy server 上傳頻寬增加的問題得到改善，而是因為過低的到達率，即是使用者

的請求銳減，而導致 proxy server 的上傳頻寬減少。

在到達率為 1 至 8 時，RR 和 OLF 的 proxy server 上傳頻寬較接近，當到達率高時，RR 可以讓所有的 description 中的 loop 維持開啟的狀態，而 OLF 主要考量 open loop 為優先加入，因此 proxy server 上傳頻寬皆維持最低。

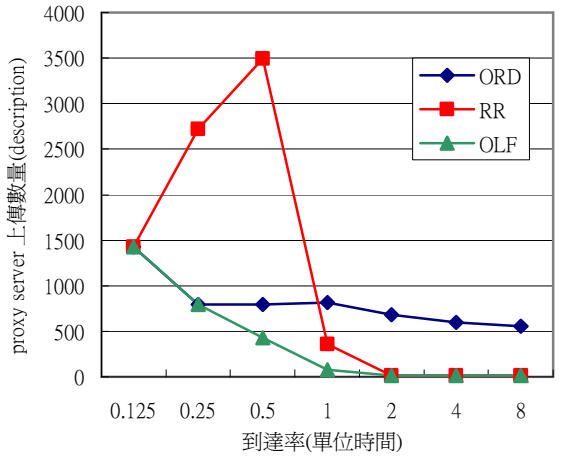


圖 15 不同到達率之 proxy server 上傳頻寬的比較

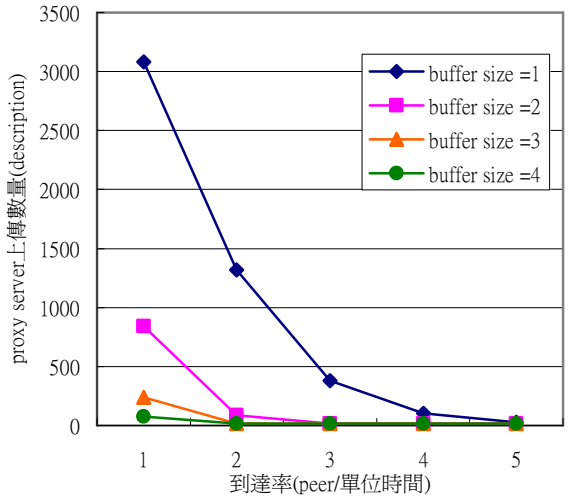


圖 16 buffer size和到達率對OLF的影響

圖 16 評估 buffer size 和到達率對 OLF 的影響中，OLF 方法用四種不同使用者 buffer 長度大小。buffer size 為 1，其長度代表 1 個時間單位。在圖中可得知，當到達率越高時，所需的 proxy server 上傳頻寬越小，因為 description 的 loop 中的 peer，在短時間內即可有下一個 peer 加入，因此所有的 description 的 loop 皆維持開啟的狀態，則可更加節省 proxy server 上傳頻寬。當使用者 buffer 長度越長時，，所需的 proxy server 上傳頻寬也越小，因為 description 的 loop 中的 peer 有較長的時間來等待下一個加入的 peer。

## 五、結論與未來研究

本研究利用 MDC 編碼技術，使得 loopback 機制在網路的異質性問題下，也可以進行串流傳輸，並且對編碼後的 description，以維持 loop 開啟為準則。OLF 分配考量目前網路平均的 description 需求量來進行適量的傳輸，及 loopback 結構上的堅韌度和可用度。當有 peer 中途離開系統時，降低原本 loop 發生斷裂，減少影響其他 peer 接收串流。且使維持影片的開頭在 peer 端，使得新加入的 peer 可以不必再向 server 請求資源，而節省整體的 server 負載。模擬實驗結果證實，OLF 方法優於 ORD 和 RR，主要是因為維持 loop 為開啟的狀態，且可發現足夠的 peer buffer 長度和到達率將可讓 OLF 方法達到較佳效能。

目前研究中，運用 MDC 編碼技術，當 peer 中途發生背棄時，如果無法用重複的串流資料修補，則會降低下游的 peer 持有請求 description 的數量，此時就需用 server 進行支援受影響的 peer，來補足遺失的 description。在未來研究中，我們將探討，當 peer 中途發生背棄時，受影響的 peer 可以由其他的 peer 來進行修復，用其他未接收的 description 來遞補其遺失的 description，維持 peer 所需的觀看品質，降低 proxy server 在 peer 斷裂的時候，所需付出的補償上傳頻寬。

## 五、參考文獻

[1] 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.

[2] 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.

[3] 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.

[4] Y. Cui and K. Nahrstedt, “Layered peer-to-peer streaming,” Proc. of ACM NOSSDAV, pp.162-171, 2003.

[5] 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.

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

[8] L. Juhn, and Tseng, L., “Harmonic broadcasting for video-on-demand service,” IEEE Trans. on Broadcasting, pp.268-271, 1997.

[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] 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.

[11] R. Rejaie, M. Handley, and D. Estrin, “Layered quality adaptation for Internet video streaming,” IEEE Journal on Selected Areas in Communications (JSAC) ,vol. 18, pp.2530-2543, 2000.

[12] 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.

[13] 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.

[14] S. Viswanathan and T. Imielinski, “Metropolitan area video-on-demand service using pyramid broadcasting,” ACM Multimedia Systems, vol. 4, No. 3, pp.197-208, 1996.

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

[16] Y. Cui and K. Nahrstedt, "Layered peer-to-peer streaming," Proc. of ACM NOSSDAV, pp.162-171, 2003.

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

[18] 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.

[19] 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.

[20] 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.