以節點分級與備援來優化樹狀拓樸下的點對點串流直播系統

Chun-Hsien Lu（呂俊賢）and Bo-wen Lin（林博文）
Dept. of Computer Science and Information Engineering
Fu Jen Catholic University
輔仁大學資訊工程系
jonlu@csie.fju.edu.tw

Abstract Today many live broadcast platforms have been widely used, resulting in a large increase in network traffic. These broadcast services began to operate under the peer-to-peer architecture, and most peer-to-peer live streaming systems often sacrifice delay and cost in order to achieve high stability. The focus of live streaming is the real-time interaction between the viewers and the live broadcaster or the real-time information of the live content. In this work, we propose a dynamic topology maintenance strategy on a tree topology to control the system architecture and reduce transmission costs. Moreover, we also use the buffer overlap decision mechanism to reduce the delay of live streaming, and the peer grading strategy to improve system stability and reduce the impact when an upstream peer gets offline suddenly. In addition, for each peer we assigned a backup peer to provide audio and video clips for that peer when it lost the upstream transmission peer. In this way, we enable each peer to receive audio and video clips stably, reduce the delay between peers and servers, and improve the smoothness of live video playback. The experimental results show that our proposed strategy can improve the streaming quality by 2%~5%, and the delay by 15%~20% compared to ordinary peer-to-peer live streaming systems.

Keywords: peer-to-peer systems, live streaming, tree-based topology, peer backup

摘要 — 隨著雲端技術的進步和網路傳輸速度提升，許多影音直播平台也都蓬勃發展，導致網路的流量大幅增加，許多影音服務也開始採用點對點架構的方式運作。在直播系統中最重要的便是其穩定性，但一般的點對點串流直播系統大多為了追求高穩定性，卻經常忽略了延遲及成本的問題，而直播的重點在於觀眾與直播主之間的即時互動性或是得知直播內容的即時資訊。在本篇論文中，我們基於樹狀拓樸提出了一個拓樸動態維護策略，能夠簡單的控管系統架構並降低傳輸成本，我們還利用了緩衝區重疊判定機制降低直播的延遲與節點分級來降低上游節點的離線機率並提升系統穩定性，並且設定備援點來負責提供影音片段給失去傳輸點的節點。透過上述策略，我們讓每一個節點都能夠穩定的收到影音片段，降低節點與伺服器之間的延遲，提升直播影音的播放流暢性。經由實驗數據證實，我們所提出的拓樸動態維護策略和一般點對點影音串流系統相較，在串流品質上提升了 2%~5%，延遲也下降 15%~20%。

關鍵詞：點對點系統、串流直播、樹狀拓樸、備援點

1. 導論

近年來隨著雲端技術與網路傳輸的發達，人們透過網路使用串流直播服務的需求也愈來愈高，許多直播平台像是 17 直播[1]、Nonolive[2]和金剛直播[3]也都隨之興起。根據大型遊戲直播平台 Twitch[4]統計，觀眾觀看在線視頻總時間從 2017 年的 3550 億分鐘成長到 2018 年的 4340 億分鐘[5][6]。而網路系統公司 Cisco 預測， 2021 年時直播將佔數據中心內總流量的 10%，影片將佔數據中心和終端使用者間傳輸總流量的 85%[7]，由此可知串流直播服務已經成為生活中不可或缺的一部分，因此提升串流直播服務的整體系統效能也有一定程度的重要性。串流直播服務架構又可分為主從式架構(Client-Server Architecture)與點對點架構(Peer-to-Peer Architecture，簡稱 P2P)，主從式架構是由一個或多個伺服器來提供用戶端直播串流，因為內容皆由伺服器端管理，不易被竄改因此安全性較高。此架構易於管理、維護與設定，但擴展性低。隨著用戶的增加，總流量也隨之成長，成本也會快速上升。而在點對點架構下，所有節點都需要同時扮演伺服器與用戶端的角色，進行上傳與下載的動作。藉由利用每一個節點的頻寬，可大幅降低主要伺服器的流量負擔及成本，且可避免因單一伺服器節點離線而導致所有影音片段傳輸失敗。在點對點串流直播系統中，每一個節點緩衝區都儲存數個影音的片段，且皆須分享影音片段給其他節點。在此架構下節點能依自己的需求選擇鄰居互相連結，所形成之拓樸又可分為樹狀、網狀及混合式拓樸。在本論文中，我們將探討的重點放在使用樹狀拓樸的點對點串流直播系統上，如圖 1 所示。

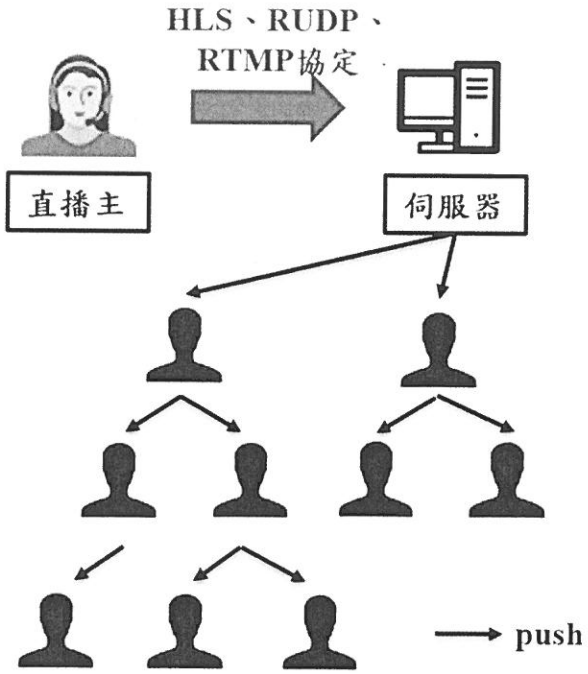

圖 1、點對點串流直播系統之樹狀拓樸

節點在點對點串流直播系統中所做的每一個動作，皆會對整體系統效能造成影響，在此我們列出一些議題如下：

- 上游節點選擇：上游節點的不同會影響自身的串流品質、延遲及成本，選擇適合的上游節點可以讓使用者順利的接收到所需要的影音片段並擁有低延遲和低成本等優點，而不適合的上游節點則讓使用者的觀賞體驗變差。
- 用戶離線狀況：節點重新進入系統或是離開系統都視同離線狀況發生，若是節點在系統中停留時間較長，則傳輸影音片段的過程中較不會因為上游節點離線而中斷。
- 節點頻寬大小：節點的頻寬大小決定此節點可以服務多少其他節點，系統內頻寬愈多則可服務的節點愈多。

上述議題皆會對系統效能產生相當程度的影響，若無法完善的管理，則系統無法提供使用者的良好觀賞品質。在現今的社會中，人手一支手機已經非常普遍，且網路速度的提升，對於點對點架構的發展非常有利。當用戶正在收看點對點直播時，也會貢獻他們裝置的頻寬，整體

系統能容納的人數多寡取決於這些用戶的頻寬總量。而在樹狀拓樸下，整體系統結構單純，一個節點只需負責其子節點的影音片段需求，且延遲較短、傳輸成本也較低。樹狀拓樸雖然達到直播低延遲的要求，也不需消耗太多的計算資源，但當上游節點離線時，樹狀拓樸若無法及時的修復，那對於整體系統的穩定性會造成極大的影響，導致大多數的子節點無法收到影音片段，最終這些節點可能選擇離開系統。這樣的結果我們並不樂見，所以在本論文中，我們希望對節點做分級的動作，依據其級別的不同，尋找適合的上游節點，我們同時對每個節點新增一個備援節點，使得當上游節點離線狀況發生時，子節點暫時由備援節點提供直播片段，因而可以有更充足的緩衝時間尋找新的上游節點，如此能夠提升影音播放的流暢度與整體系統的穩定性。在本論文中我們提出上游節點選擇方法與系統節點管理方法，透過這兩種方法來改善樹狀拓樸直播系統的效能。

本論文其餘部分內容如下，第二節將介紹點對點串流直播系統的相關研究，第三節說明我們所提出的方法，第四節為效能評估，第五節則是結論。

2. 相關研究

[8-11]中的作者們皆有探討鄰居節點連接方式的議題，[8]為了追求直播的穩定性，選擇停留時間較長的節點，使這些節點互相連接，形成系統中的核心區域，長時間的為系統服務，提供系統中其他節點的影音片段。[9]也利用穩定的節點來為系統服務，而為了降低節點之間的傳輸延遲，節點在選擇鄰居時會以 ISP 相同或較近的節點為鄰居。[10]則探討同 ISP 內與跨 ISP 的節點連接方式，若節點之間所跨越的 ISP 愈多，則傳輸延遲會隨之增加。因此該作者將節點依據 ISP 做分區，相同 ISP 的節點之間會優先互相選擇為鄰居。而每個 ISP 會挑選出部分節點作為超級節點，超級節點對內負責保持自己 ISP 區域內的穩定性，對外與其他 ISP 的超級節點連接，作為兩個 ISP 之間溝通的橋樑，這其中對外的節點若擁有高頻寬的優勢則會被提升為核心節點。[11]則屬於 CDN-P2P 混合式架構，利用 Tracker 紀錄系統拓樸、伺服器與節點資訊，並提供適合

成為鄰居的節點清單給予新加入的成員。當新的成員進入系統時，Tracker 會選擇兩個距離最近的 CDN Server，在此兩個 CDN Server 所服務的節點會計算他們與新加入的節點間的最短路徑，最後以平均最短路徑作為選擇 CDN Server 的依據。而在新成員加入 CDN Server 後，則選擇距離較近的節點作為鄰居。作者利用 Tracker 運算，提供最適合連接方式給新成員，以此方法可以大幅地提升系統效能。

在節點的分級方式方面，[8][9]以頻寬高低與節點停留時間作為判定其在系統中階級的依據，在拓樸建立時將不同階級的節點分配至不同區域以提升系統穩定性。[12]提出一套節點選擇演算法，當一個新的節點加入時，會從系統中挑選 10 個節點做為候選節點，計算其傳輸速率與延遲，得出候選節點效能，並將適合的候選節點放入備用集合，接著系統會啟用主要集合管理器及備用集合管理器。由於一開始主要集合並無節點，主要集合管理器會向備用集合的數個候選節點詢問是否能夠建立連線，若成功則將節點放入主要集合。而備用集合管理器負責在備用集合內的候選節點不足時，在系統中尋找新的候選節點。當主要集合的節點發生問題，主要集合管理器會從備用集合挑出數個的候選節點建立連線，以維持播放流暢性。

在點對點串流直播系統中，串流品質、傳輸成本及延遲都是非常重要的效能指標，因此我們希望透過結合上述多種方法，並在樹狀拓樸的架構下，提出一個演算法，善用樹狀拓樸的優點並提升樹狀拓樸的穩定性，以增加觀眾收看直播品質來改善整體系統效能。

3. 方法設計

3.1 系統假設

直播可以播放多種類型的影片，像是比賽、遊戲和演唱會等等。觀眾收看直播時，會選擇感興趣的類型，且興趣愈高則停留時間愈長，甚至有些觀眾會收看完整場直播。這代表隨著停留時間愈長，觀眾的離線機率愈低。我們認為停留時間長的觀眾對於穩定系統的幫助很大，因此我們以停留時間較長的節點作為整個系統的核心，並儘量指派他們擔任新加入節點的上游節點，

以維持直播的穩定性，服務更多對這場直播有興趣的觀眾。在我們的策略中，上游節點包含傳輸點和備援點，傳輸點為主要提供影音片段的節點，備援點則是傳輸點無法正常提供影音片段時的替代節點。系統中所有節點都有機會擔任傳輸點或是備援點，藉以提升整體系統的穩定性。每一個節點都擁有緩衝區，分為三個部分，分別為播放點、播放點前的已播放區和準備播放的預存區，如圖2所示。已播放區為已收看並準備推送給下游節點的片段，播放點為當前收看之片段，而預存區為準備播放之片段，且可作為當節點需要重新尋找傳輸點和備援點時的預留時間。

圖 2、緩衝區結構

我們假設直播開始時，新的節點會陸續進入，Tracker 會記錄這些新的節點，並參考已存在系統中節點的緩衝區內容以及其他參數，來替新的節點分配合適的上游節點。上下游關係建立完成後，傳輸點會開始傳輸影音片段，新的節點收到影音片段後便可觀看直播。若傳輸點或備援點因離線無法提供片段，下游節點再通知 Tracker 重新尋找合適的上游節點，直到節點選擇離開直播或是直播結束，系統流程圖如圖 3 所示：

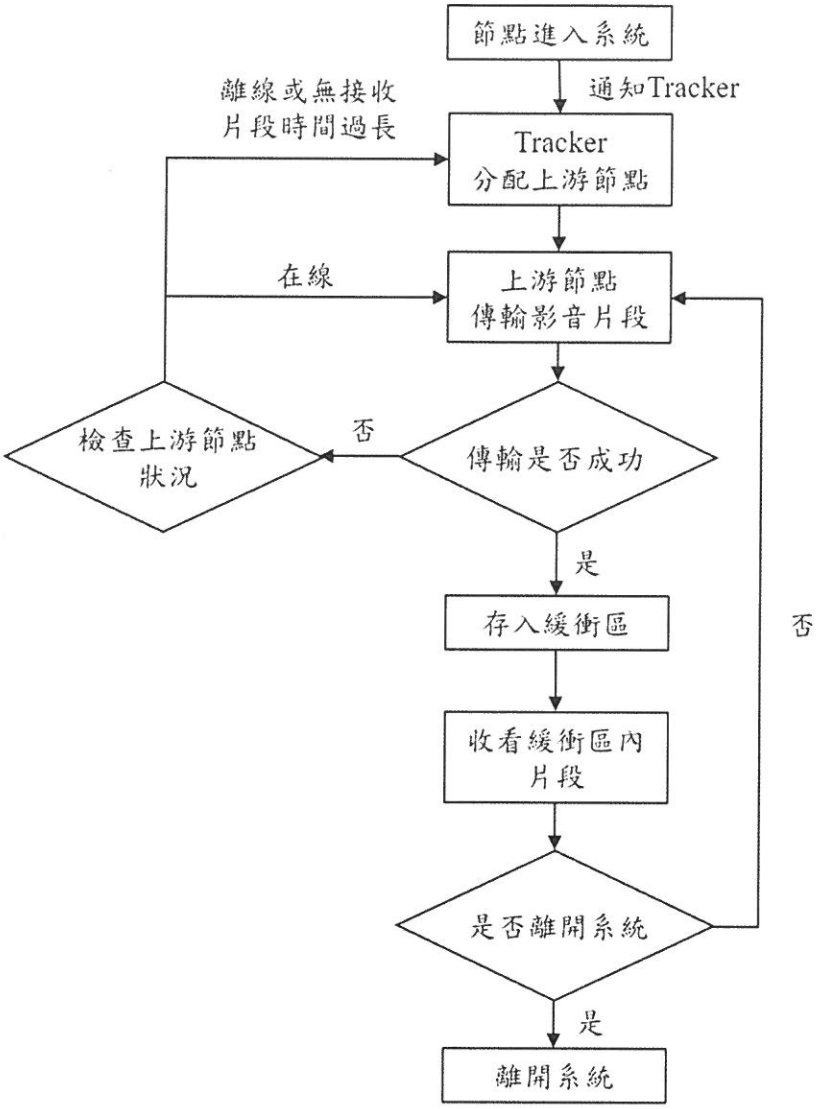

圖 3、系統流程圖

3.2 系統節點管理方法

Tracker 扮演系統中最重要的角色，負責節點分級、分配上游節點和維護拓樸等工作。所有節點的加入與離開都會被 Tracker 紀錄，並且每隔一段時間需要通知 Tracker，確保其在線的狀態。若節點自身的狀態改變也須向 Tracker 傳遞新的資訊，Tracker 會即時更新其節點狀態表。 我們假設停留時間較長的節點往後的離線機率也會較低，因此我們將節點以停留時間由長到短分為 A、B 和 C 三級，且節點有升級機制，升級門檻以停留時間來做為升級的依據。如圖 4 所示，假設節點升為 B 級的門檻為 T2，升為 A 級的門檻為 T4，在 T1 至 T4 之後的這段期間，其離線率

每過一段時間便會降低，此時有一個新的節點 N 加入系統，只要 N 的停留時間達到 T2、T3 和 T4 其離線率便會降低，而達到 T2 系統會將節點升為 B 級，T4 則升為 A 級。其中 A 級代表系統中最穩定的節點，離線機率最低，在其之下的子節點也最多，當其離線時對系統影響最大。B 級則是已在系統內待一段時間但並不算長，所以 B 級以下的子節點不會太多。C 級則是剛進入系統的節點，離線機率假設最高，除非 A、B 級節點的總頻寬不足，否則 C 級將不會成為上游節點。

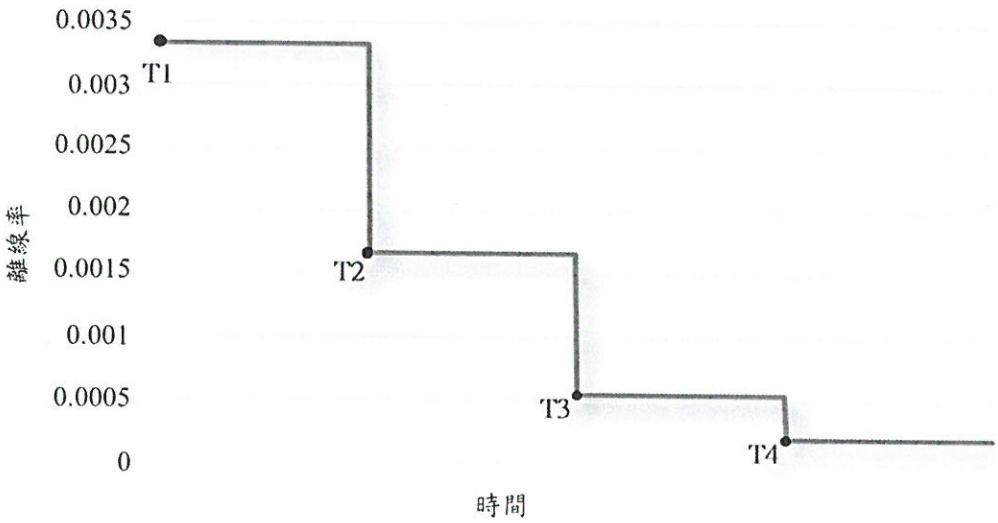

圖 4、節點離線機率圖

3.3 上游節點選擇方法

系統中所有節點都是由 Tracker 統一管理，當新的節點加入後，Tracker 首先會將節點歸類為 C 級，並優先挑選某 A 級節點為其傳輸點。假如 A 級節點總頻寬不足，則挑選 B 級節點，若 B 級總頻寬不足則挑選 C 級節點做為傳輸備援點，是依據節點的傳輸點級別來做挑選，若傳輸點為 A 級，則可依照 B、C、A 三級的順序挑選另一節點作為備援點；若傳輸點為 B 級，則依照 C、B 的順序挑選兩種級別的節點作為備援點；若傳輸點為 C 級，只能夠選擇 C 級節點作為備援點。在傳輸點改變時，備援點也會依上述規則做更換。上述規則降低了 A 級節點被選擇為備援點的

機率，有效的保留 A 級節點的總頻寬，讓大部分 A 級節點擔任其他節點的傳輸點，達到大部分節點一開始就能夠擁有良好串流品質的目標，而 B 級節點和 C 級節點大多數擔任備援點，作為傳輸點離線時的串流提供者，以穩定系統運作。傳輸影音片段時，傳輸點假設皆會傳輸高品質的影音片段，因此需付出較多的頻寬，備援點則因為需要擔任多個節點的替代節點，會為下游節點保留最低限度的頻寬，因此在下游節點有需求時，能夠依據保留頻寬提供適當品質的影音片段。假設節點 P1 需要一個備援點，由於它的傳輸點是 A 級節點，因此 Tracker 會先從 Level B 開始挑選頻寬足夠且非 P1 之子節點的目標 P2，接著再將 P1 與 P2 兩者之間建立起上下游關係。以此類推，其他節點也是以相同的方式建立上下游關係。整體系統架構如圖 5 所示：

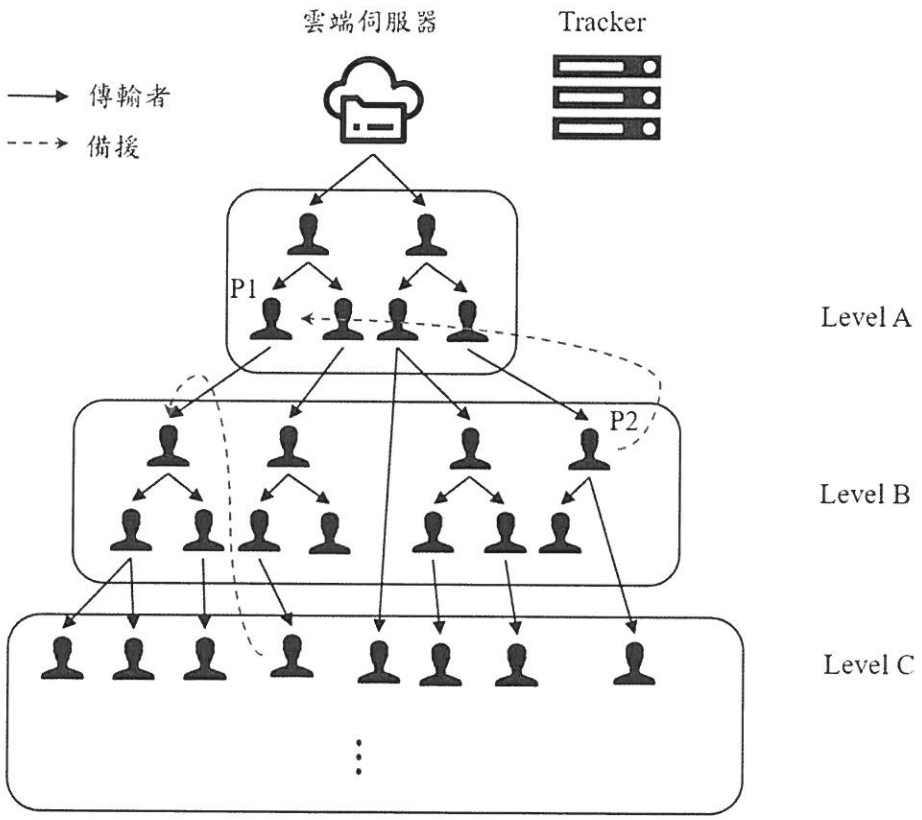

圖 5、系統架構

上游節點的選擇除了以級別區分外，還需確認上游節點緩衝區內有無下游節點所需片段。例如：上游節點緩衝區可存放5秒影音片段，則下游節點會依據最新接收到的片段與上游節點的緩衝區片段重疊部分來決定是否建立連線。以圖 6 為範例，一格代表一秒鐘的片段，Node 4 希望下一個收到的片段為 6，所以 Node 2 或 3 皆可擔任其上游節點。Node 2 希望下一個收到的片段為 9，所以 Tracker 分配 Node 3 為其上游節點。以此類推，Node 1 會被分配為 Node 3 的上游節點。由於系統中並無節點適合作為 Node 5 的上游節點，若分配 Node 4 為其上游節點，則 Node 5 所接收到的影音片段會和伺服器的影音片段差距較大，延遲也高達 11，進而拖累整體系統效能，若選擇 Node 1 為其上游節點則與伺服器之間差距較小，延遲最多也只有 5，因此 Tracker 會優先分配延遲最低的 Node 1 為其上游節點。

伺服器	Now=15				
Node 1	10	11	Now=12	13	14
Node 2	4	5	Now=6	7	8
Node 3	6	7	Now=8	9	X
Node 4	Now=4	5	X	X	X
Node 5	Now=0	X	X	X	X

圖 6、上游節點緩衝區選擇

系統中每一個節點都有可能隨時離開系統，當某一節點離開系統時，其上游節點會失去傳輸影音片段的目標，而其下游節點可能會接收不到所需要的影音片段。此時已離線節點的上下游節點由於狀態改變，所以皆會向 Tracker 通知更新其節點狀態表，Tracker 收到通知後會開始修復拓樸，修復規則和替新加入的節點尋找上游節點相同。在 Tracker 修復拓樸的期間，傳輸點離線的節點，其影音片段暫時由備援點提供，直到與另一組新的上游節點成功連接為止。

4. 效能評估

4.1 實驗環境

我們使用 Java 語言撰寫一個模擬程式來評估我們所提出的方法效能，比較對象為同樣使用樹狀拓樸的串流直播系統，包含只使用節點分級但未使用備援點的方法，以及兩者皆未使用的方法。在模擬實驗中，我們假設節點的下載頻寬皆預設為足夠接收高品質的影音片段。直播模擬總時長為 7200 秒，每 2 秒會有一個新的節點加入，且每一個節點每一秒鐘都有可能會離線，因此我們將在不同時間點設定門檻值，分別為 T1：0 秒、T2：300 秒、T3：600 秒與 T4：1200 秒，在 T1 至 T2、T2 至 T3、T3 至 T4 和 T4 以後的離線率分別為 1/300、1/600、1/1800 和 1/4800，且 T3 和 T4 為升為 B 級和 A 級的門檻。

所有直播片段皆由伺服器提供，每個片段可播放一秒鐘，直播結束節點就離開系統。本實驗中我們配置一個伺服器、一個 Tracker 和 3600 個節點，節點緩衝區大小可以保存 5 秒鐘的影音片段，節點重新取得上游節點的時間設定為快速 3~5 或慢速 8~10 秒，系統拓樸會隨著節點的加入與離線而變動。伺服器的上傳頻寬為 30Mbps，而節點的上傳頻寬分為 6Mbps、12Mbps 及 18Mbps 三種情形。直播影片共包含 7200 個影音片段，分為高階品質 1080p、中階品質 720p 及低階品質 480p，三種品質所需上傳頻寬分別為 6Mbps、2Mbps 及 1.2Mbps。傳輸點所傳輸的影音片段皆為高階品質，而備援點會依據所剩頻寬及所需服務下游節點數量來決定所要傳輸的影音片段品質。例如：節點 A 有 12Mbps 的上傳頻寬，且已成為一個節點的傳輸點(占用上傳頻寬 6Mbps)及五個節點的備援點(保留上傳頻寬 1.2Mbps * 5)。假設此時 A 有三個下游節點因為傳輸點無法提供影音片段，需要由 A 這個備援點提供影音片段，則 A 需要使用保留的頻寬，分別傳輸三種相同品質的片段給這三個節點，所以 A 會選擇提供 3 份中階品質的片段給這三個節點，詳細參數如表 1 所示。

表 1、模擬實驗參數設定

伺服器數量	1
Tracker 數量	1
伺服器上傳頻寬	30Mbps
節點上傳頻寬	6Mbps、12Mbps、18Mbps
模擬時間	7200 秒
T1、T2、T3、T4	0 秒、300 秒、600 秒、1200 秒
離線率	T1 至 T2：1/300、T2 至 T3：1/600、T3 至 T4：1/1800、T4 以後：1/4800
影片所需頻寬	6Mbps、2Mbps、1.2Mbps
節點緩衝區大小	5 個片段
節點進入間隔時間	2 秒
重新取得上游節點時間	快速 3~5 秒/慢速 8~10 秒

4.2 實驗結果

在本實驗中，我們在相同的模擬環境中比較三種方法的節點收到的影音串流品質、延遲、hop 數與備援點使用率。Pro 為我們所提出的策略、NB 為只做節點分級選擇上游節點但並未使用備援點的策略、NLB 為未做節點分級和備援點的策略，僅以尋找延遲最低的節點作為上游節點。方法後的標示若為 H 則代表此方法系統中節點上傳頻寬皆為 18Mbps，為比例則代表系統中節點上傳頻寬為 6Mbps 和 12Mbps 的數量比。各項效能指標定義如下：

- 串流品質：計算節點收到的影音片段品質，所有節點預設為都應接收到高階品質的影音片段，若因某些原因導致只收到中階品質或低階品質的影音片段，甚至是沒有收到影音片段，皆會造成串流品質的下降。

- 延遲：節點目前的播放點片段與伺服器目前送出最後一個影音片段的時間差。
- Hop 數：伺服器與節點之間須經過多少個節點。
- 備援點使用率：我們提出的方法中，備援點在系統中被使用的比率。

圖 7 至圖 9 中，節點上傳頻寬可分為 6 Mbps、12Mbps 及 18Mbps 三種，不同頻寬的節點以不同的數量比例進入系統，Tracker 重新分配上游節點所需時間為 8~10 秒。隨著直播開始，我們每 6 分鐘對所有節點收到之平均串流品質做一次統計。當系統剛開始時，多數的節點皆呈現不穩定的狀態，大多數影音片段皆由伺服器提供，系統總頻寬不足，且節點離線的情況時常發生。當直播進行一段時間後，如圖 7 所示，在所有節點上傳頻寬皆為 18Mbps，系統總頻寬極度充足的情況下，雖然 NB 與 NLB 的串流品質最後也能趨於穩定，但 PRO 一開始便擁有穩定的串流品質，因此我們認為 PRO 能夠在直播剛開始時留住更多的觀眾。圖 7 中我們假設所有節點上傳頻寬皆為 12Mbps，頻寬勉強充足的情形下我們可以觀察到，在前 36 分鐘時 NB 與 NLB 皆處於不穩定的狀態，尤其在前 24 分鐘系統節點數量不多時，只要有節點離線，對串流品質的影響就非常的大，所以串流品質表現得較差，直到 42 分鐘後才趨於穩定狀態。而 PRO 因為有備援點的緣故，在系統最不穩定的時候，節點的串流品質相較於 NB 與 NLB 兩種方法還可以高出 5%左右，且系統也能夠較快速地趨於穩定。圖 9 顯示當系統頻寬較低時的情況，和圖 7 和 8 相比，PRO、NB 與 NLB 需要使系統趨於穩定的時間皆拉長。但直播剛開始時的串流品質對於觀眾去留的影響非常大，趨於穩定的時間需要愈長，觀眾愈容易離開，而 PRO 雖然受到影響，但卻不至於如同對 NB 影響來得大。PRO 的串流品質也因為備援點存在的關係，在系統處於穩定狀態時，串流品質依然比 NB 高出 1%~2%，在系統不穩定時，甚至高出 5%左右。且由 NB 與 NLB 相比較可以發現，節點分級也為串流品質帶來了 1%左右的提升。

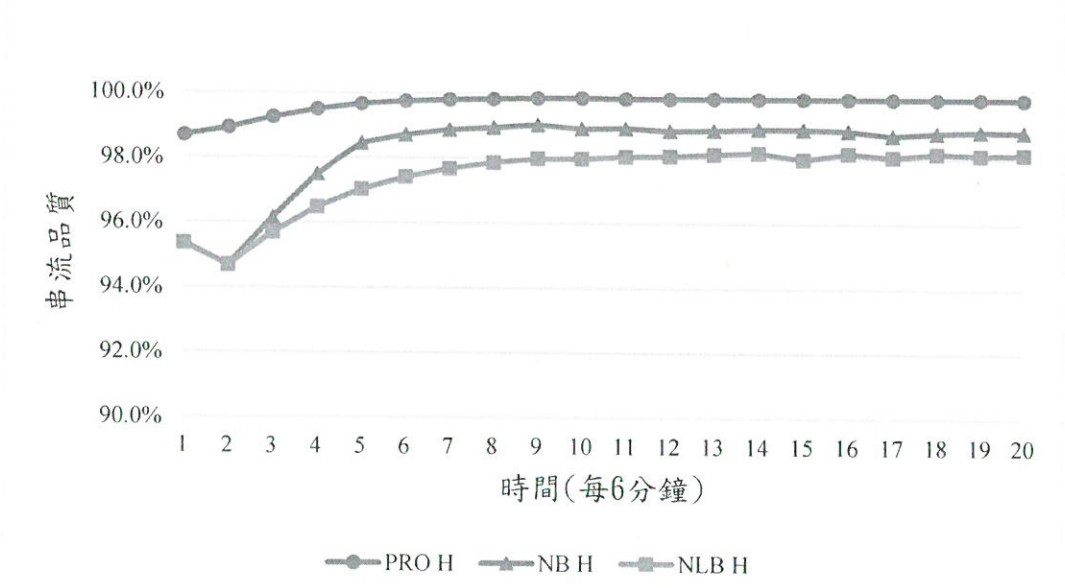

圖 7、串流品質比較圖(節點上傳頻寬皆為 18Mbps、重新分配上游節點時間 8~10 秒)

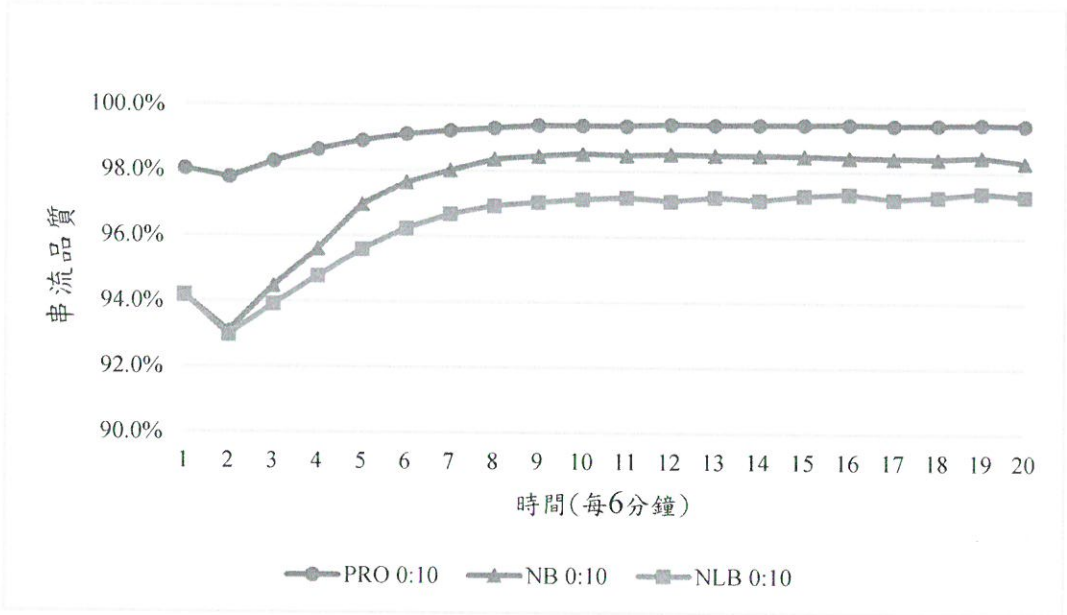

圖 8、串流品質比較圖(節點上傳頻寬 6Mbps 與 12Mbps 數量比為 0:10、重新分配上游節點時間 8~10 秒)

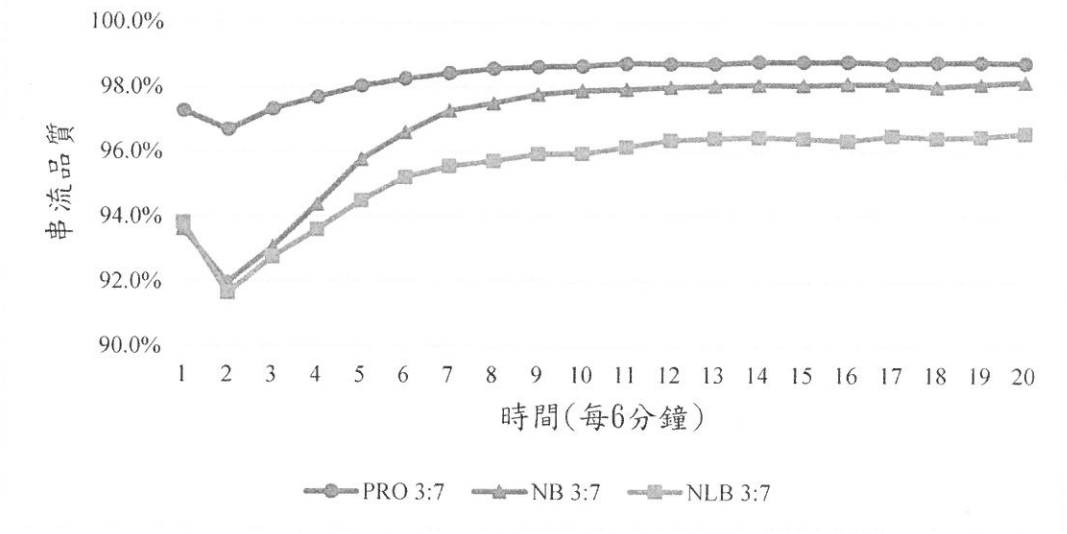

圖 9、串流品質比較圖(節點上傳頻寬 6Mbps 與 12Mbps 數量比約為 3:7、重新尋找分配節點時間 8~10 秒)

圖 10 至圖 12 分別對應到圖 7 至圖 9 的環境參數。我們在三種方法中皆使用緩衝區重疊部分做為 Tracker 重新分配上游節點的參數，降低了節點與其上游節點之間的延遲差距。從圖 10 至圖 12 我們觀察到，在系統總頻寬充足的情況下，NB 的延遲會稍微高於 NLB，是因為 NLB 在緩衝區重疊判定結束後，是以延遲來做為選擇上游節點的第二種依據，而 NB 則是以高級別節點做為選擇上游節點的依據以求取系統的穩定性，所以延遲略高。PRO 由於有備援點的幫助，在上游節點離線的情況發生時，依然能夠從備援點收到影音片段，直到重新與新的上游節點成功配對。此機制讓緩衝區能夠持續收到影音片段，降低因為離線所造成的負面影響。但 NB 與 NLB 沒有備援機制，雖然緩衝區能夠輔助降低延遲，但在上游節點離線時，仍有 8~10 秒鐘收不到片段，導致延遲大幅上升。

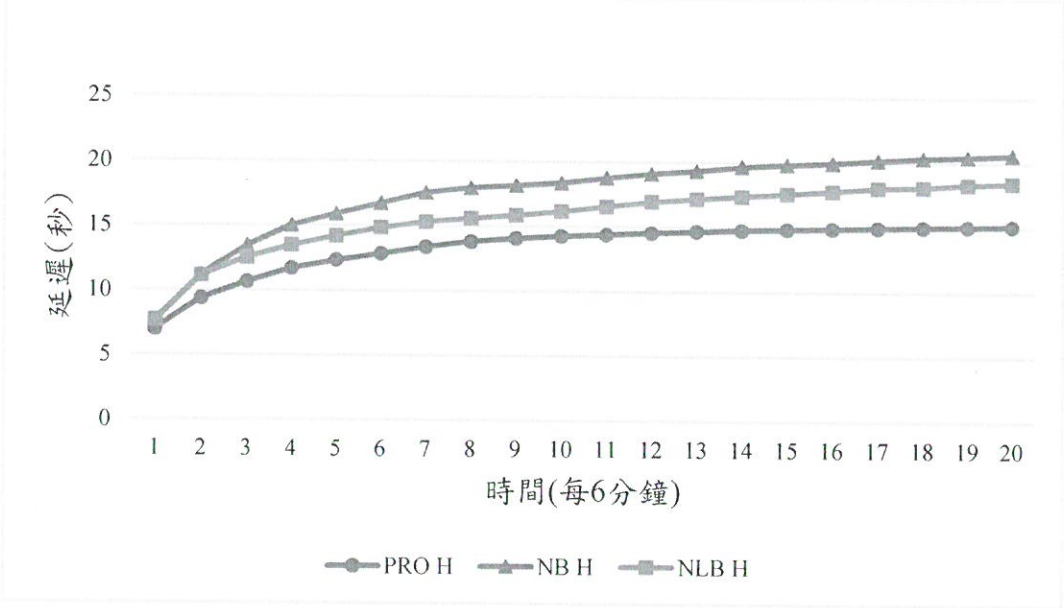

圖 10、延遲比較圖(節點上傳頻寬皆為 18Mbps、重新分配上游節點時間 8~10 秒)

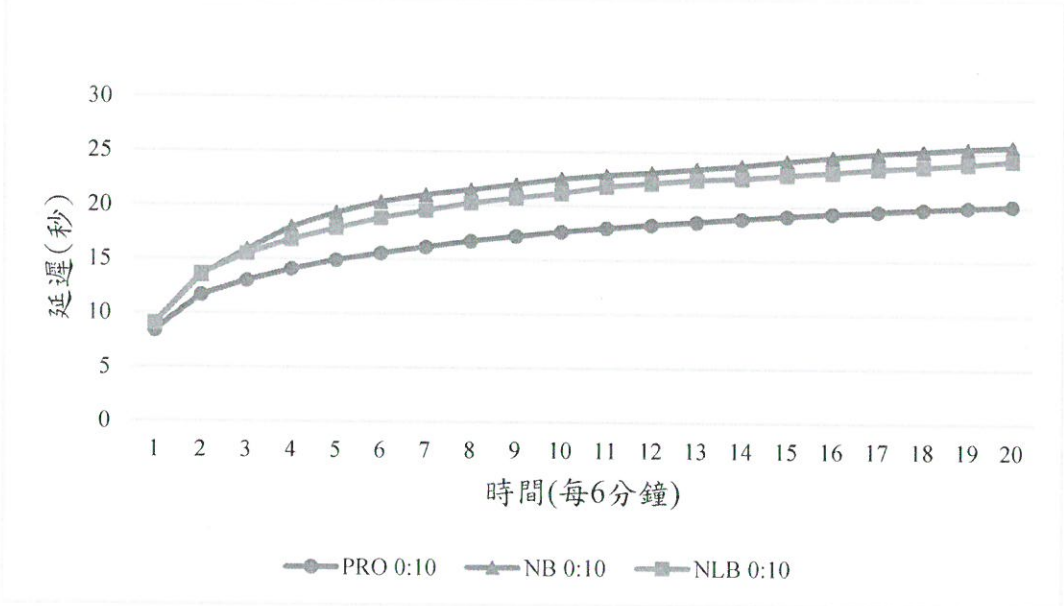

圖 11、延遲比較圖(節點上傳頻寬 6Mbps 與 12Mbps 數量比為 0:10、重新分配上游節點時間 8~10 秒)

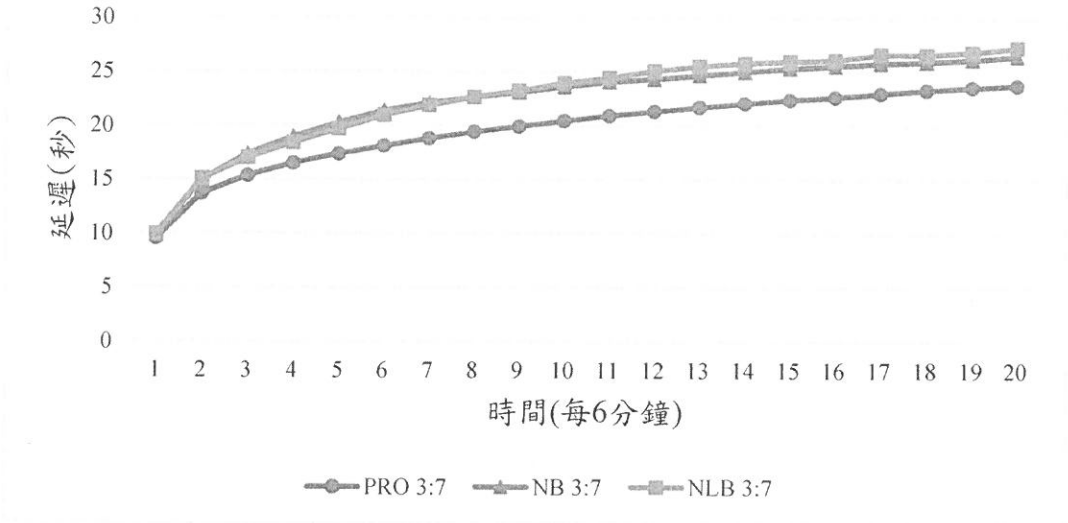

圖 12、延遲比較圖(節點上傳頻寬 6Mbps 與 12Mbps 數量比約為 3:7、重新分配上游節點時間 8~10 秒)

我們從圖 13 觀察到，在系統頻寬不足的情況下，若是我們縮短 Tracker 重新分配上游節點所需時間，短時間內處於重新尋找上游節點狀態的節點數量較多，它們之間也較容易發生互相搶奪上游節點的情況，造成部分節點只能尋找低級別節點做為上游節點，但低級別的上游節點較不穩定，所以更容易導致串流品質下降。且在 PRO 中，因為需要保留部分頻寬給備援點，所以當系統總頻寬不足以分配給新加入的節點或上游節點剛離線的節點時，系統內有收到影音片段的節點就會減少，能提供影音片段的節點也跟著減少，造成 PRO 的串流品質下降，而 NB 與 NLB 兩種方法因為系統總頻寬需求較低，所以只需維持平均每一個節點能夠分配到 6Mbps 便能夠穩定的運行。

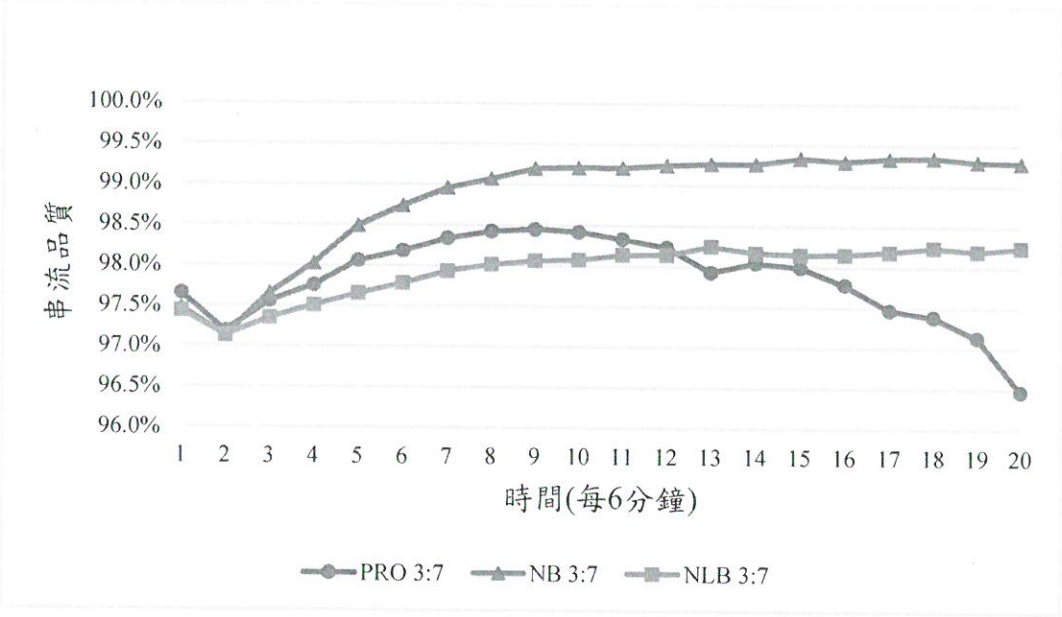

圖 13、串流品質比較圖(節點上傳頻寬 6Mbps 與 12Mbps 數量比約為 3:7、重新分配上游節點時間 3~5 秒)

圖 14 對應到圖 13 的環境參數。由圖 13 可以得知，PRO 在系統內能夠成功收到影音片段的節點減少，因此無法收到影音片段的節點延遲會慢慢被拉大，直到成功被重新分配新的上游節點或是得到新的影音片段。如圖 14 所示，PRO 的延遲漸漸增加，而 NB 與 NLB 皆優於 PRO。

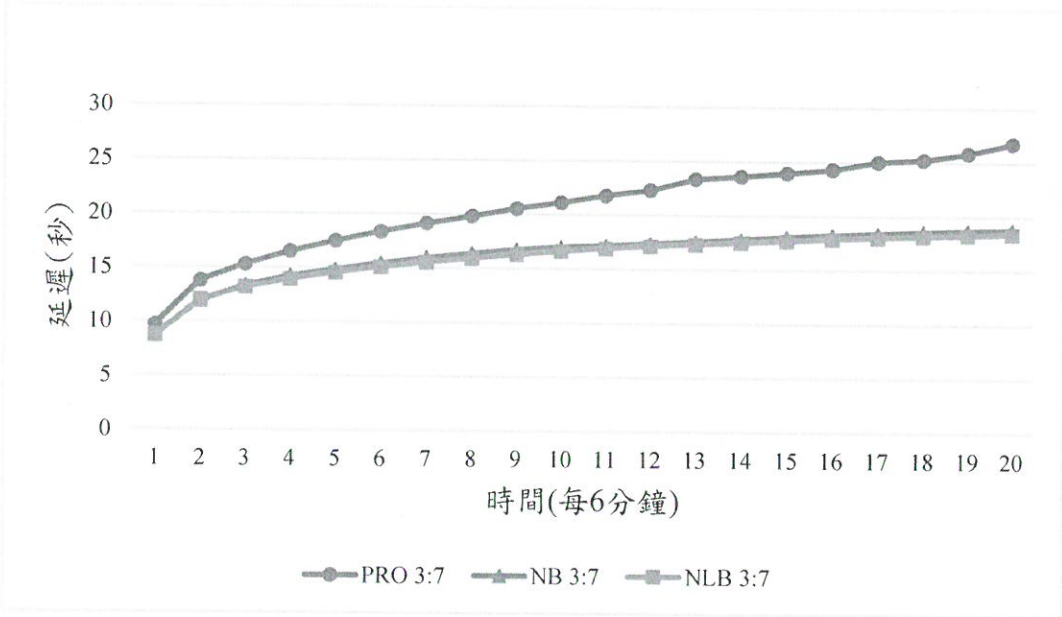

圖 14、延遲比較圖(節點上傳頻寬 6Mbps 與 12Mbps 數量比約為 3:7、重新分配上游節點時間 3~5 秒)

圖 15 顯示備援點在不同系統頻寬下被使用的狀況，我們統計節點從傳輸點與備援點收到片段的總比例，我們可以發現在系統總頻寬越低時，系統內能穩定提供影音片段的節點越少，所以備援點能夠提供的幫助就越大，且能夠有效的提升使用者收看直播時的播放連續性。

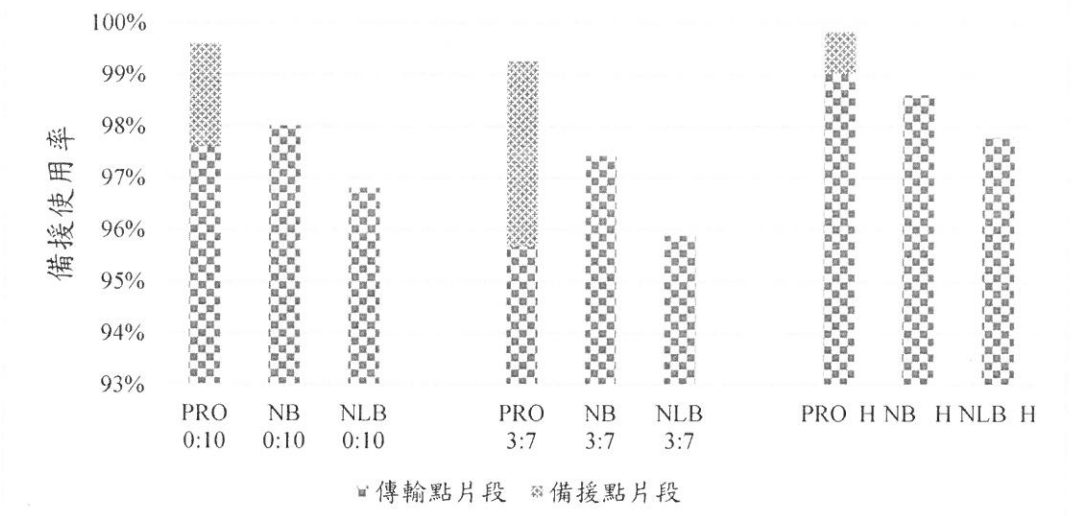

圖 15、不同系統頻寬下的備援點使用率比較圖(重新分配上游節點時間 8~10 秒)

5. 結論

雖然雲端技術與網路頻寬快速的發展，但主從式架構依舊面臨著成本及擴展性的瓶頸，因此許多串流直播服務開始採用點對點架構，試圖擺脫主從式架構的缺陷。一般的點對點串流直播系統大多為了追求高穩定性，經常忽略了延遲及成本，而直播的主要目的為觀眾與直播主之間的即時互動性或是得知直播內容的即時資訊，因此成本、穩定性與延遲皆同等重要。

為了解決上述問題，我們提出了一個拓樸動態維護策略，期望在成本、穩定性與延遲之間達到一個平衡。我們選用控管系統架構較為簡單的樹狀拓樸來達成降低成本的目的，使用節點分級及備援點維持影音片段的傳輸連續性來提升系統穩定性。我們還利用緩衝區的影音片段連續性及選擇上游節點時以緩衝區重疊的判定機制，來成功降低節點與節點之間的延遲。

經由實驗數據顯示，在串流品質及延遲這兩項指標我們的方法都有顯著的改善。相較於NLB，串流品質上我們有 2%~5%的提升，延遲也下降了 15%~20%。此外，我們還對節點上傳頻寬及 Tracker 重新分配上游節點的時間做變動，並可以從中觀察得知，我們提出的方法在系統總頻寬足夠的情況下，能夠明顯的改善系統的效能。

參考文獻

[1] 17 直播, https://17.live/

[2] Nonolive, https://www.nonolive.com/

[3] 金剛直播, https://www.kingkong.com.tw/

[4] Twitch, https://www.twitch.tv/

[5] Twitch 2017 Year in Review Reveals 22% Growth in Watch Time, https://esportsobserver.com/twitch-2017-in-review-milestones/

[6] The State Of Twitch In 2018, https://kotaku.com/the-state-of-twitch-in-2018-1831106882

[7] 思科全球雲端指數預測 2021 年雲端流量將佔數據中心總流量的 95%, https://www.cisco.com/c/zh_tw/about/news-center/news-20180212.html

[8] F. Wang, Y. Xiong and J. Liu, "mTreebone: A Hybrid Tree/Mesh Overlay for Application-Layer Live Video Multicast," *27th International Conference on Distributed Computing Systems (ICDCS '07)*, 2007.

[9] K. Pal, M.C. Govil and M. Ahmed, "A New Hybrid Approach for Overlay Construction in P2P Live Streaming," *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 431-437, 2015.

[10] M. Liu, X. Ma, X. Luo, F. Lu and Z. Qin, "An ISP-Friendly Hierarchical Overlay for P2P Live Streaming," *2015 IEEE Global Communications Conference (GLOBECOM)*, pp. 1-6, 2015.

[11] A. Saengarunwong and T. Sanguankotchakorn, "A Two-Step Server Selection in Hybrid CDN-P2P Mesh-based for Video-on-Demand Streaming," *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 499-504, 2018.

[12] R. Liao, S. Yu and L. Dong, "A Novel Peer Management Model for Live Peer-To-Peer Streaming," *2007 International Symposium on Intelligent Signal Processing and Communication Systems*, pp. 538-541, 2007.