多資料源及快速傳送樹回復之應用層群播方法

黃家煇 柯開維 吳和庭 吳俊廷
國立台北科技大學 國立台北科技大學 國立台北科技大學 國立台北科技大學
t5599006@ntut.edu.tw kwk@csie.ntut.edu.tw htwu@csie.ntut.edu.tw t8598043@ntut.edu.tw

***摘要*—目前應用層群播的研究都注重於單資料來源群播之研究，而無法有效率的使用於有多資料來源 、動態群組調整及快速傳送樹回復等要求的應用中。本論文之目的在於提出一個適用於多資料來源並且具有快速傳送樹回復能力之應用層群播方法。在本論文提出之方法中，每一個群組成員為每一個來源端之傳送樹建立部份的傳送路徑，而所有成員執行所提出之方法後將分別為各來源端建立傳送樹。此建立傳送樹的方法不但能夠避免文獻[2]之方法所可能有之單點失效及傳輸瓶頸等問題，也能夠將傳送樹回復所需之時間縮短至簡單的查表動作之時間。透過電腦模擬的驗證，本論文所提出之方法適用於多資料來源的應用，且能夠有效的縮短傳送樹回復的時間。**

關鍵詞：Multi-source multicast, Application layer multicast, Overlay network, Fast recovery, Peer-to-Peer network.

1. Introduction

應用層群播(Application layer multicast, ALM) 將群播的功能實作於應用層[1-4]的概念解決了 IP 群播因為群組之建立與管理(group state management)、擁塞控制(congestion control)、可靠性傳送(reliability)及流量控制(flow control)等需求而需要路由器支援的限制。近年來應用層群播之研究除了在資料繞送協定(Routing Protocol)方面的研究外，群組成員離開群組或失效後之傳送樹回復(failure recovery)[5-8]、可靠性群播[9-11]等主題也有相當多的研究。然而，不管是在繞送協定本身或是其相關主題之研究都是注重於單資料來源(single source)的群播應用，而無法有效率的應用在多資料來源(multi-source)的應用環境中(如 distributed game、teleconferencing、virtual classrooms 及 multimedia chat groups 等應用都屬於多資料來源之群播型態)。多資料來源應用層群播與單資料來源應用層群播不同的是，在多資料來源的應用中，每一個群組成員都可能是資料來源端，因此，若是以單來源群播的方式來為每一個來源端建立傳送樹，將會因大量的交換訊息而佔用大量的頻寬資源。

應用層群播的群組管理及傳送樹的建立是 Peer-to-Peer(P2P)系統的一種應用。所有的群播群組成員將形成一個 P2P 網路，而所有在群組中之成員都可能會在不同的時間以不同的方式離開群組(leave 或 failure)。此外，因為應用層群播和 IP 群播在群播的運作與參與成員的不同，參與應用層群播的成員同時兼俱了群播成員(群播資料的發送與接收端)與路由器的資料轉送工作(data forwarding)，因此成員一旦離開(leave 或 failure)群組都將會影響進行中的資料傳送。然而，在 distributed game 及 live media streaming 等對時間較敏感之應用中，成員以正常之群組管理所規範之方式或因失效而離開群組後，重新建立資料傳送樹的時間將會是一項影響服務品質(Quality-Of-Service, QoS)的重要考量。

本論文提出一個適用於多資料來源的應用層群播及其傳送樹回復方法。利用本論文提出

之 *transmit/forward method* 的概念，群組成員能夠根據不同來源端之資料來決定其所應轉送的下一轉送點(next hop)，而使其能適用於多資料來源的應用。此外，本論文之群組管理在群組成員之間形成之完全網狀拓撲(fully-mesh topology)使得在偵測到成員離開群組後能夠立即透過查表動作來決定新的下一轉送點而改變傳送方式進而縮短傳送樹回復的時間。

本論文的架構如下：第二節描述應用層群播繞送協定及傳送樹回復方法的相關研究，第三節描述本論文所提出之多資料來源應用層群播及傳送樹回復之方法，第四節為模擬結果之探討，第五節為結論。

2. Related Works

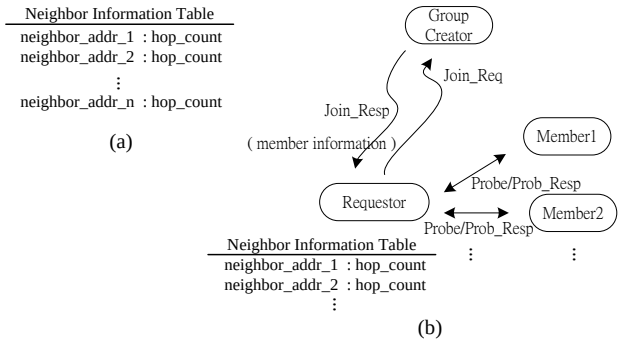
目前多資料來源應用層群播之研究注重於將群播實作於結構化(structured) P2P 網路上[15-16]。然而，結構化 P2P 網路通常都是採取 distributed hash table (DHT)的方式來建立群組成員間的連接方式及網路資源之分存放位置，然而，也因為 DHT 之方式需要大量的計算來分配 P2P 網路上之各項資源之存放位置，所以維持此種 DHT-based 的多資料來源應用層群播也會產生大量的計算耗費。而另一種多資料來源群播常用的方式為 single share tree[2]，此種方法之概念為以單資料來源之應用層群播在群組中建立單一傳送樹，而所有之來源端都將群播資料傳送之此傳送樹之頂點再透過此傳送樹傳送至所有成員，此種方式因為以單一傳送樹傳送所有來源端之資料，所以頂點將會造成資料傳送時的瓶頸(bottleneck)及單點失效(single-point-of-failure, SPOF)等問題。另一方面，對於多資料來源應用層群播之傳送樹回復問題之探討，則尚未有此方面的研究提出。

單資料來源應用層群播已有相當多的方法已被提出[1-2][12-14]，而這些方法通常會將網路組成控制及資料拓撲。成員透過在控制拓撲上的訊息交換而獲得其它成員之離開或失效等訊息，資料拓樸則是用來進行資料的傳送。若是以建立控制拓撲及資料拓撲的順序來區分， 單資料來源應用層群播大致上可區分成 Mesh-first[1][11]、Tree-first[2][12]及 Implicitly[13]三種方式。在 mesh-first 的方式中，成員之間利用訊息交換會先形成一個網狀拓撲(mesh topology)，而傳送樹的建立則是在此網狀拓撲上以任何已有之 reverse shortest path 的方式建立以來源端為頂點之傳送樹，因為傳送樹的建立完全基於在先前建立之網狀拓撲上，因此，建立之傳送樹的效能將與此網狀拓撲的品質有直接的關係，所以 mesh-first 之方法通常都需要有一些增進網狀拓撲之品質(如跳躍數(hop count))的機制來動態調整成員加入及離開而造成的拓撲變動。Tree first 的方式則是直接以來源端為傳送樹之頂點而根據跳躍數或其它拓撲特性而建立傳送樹，在建立完傳送樹後，每個成員將利用隨機或特定機制找到其它群組成員並與其建立一條控制連結(control link)而形成控制拓撲。Implicitly 之方式則利用特定之拓撲特性(延遲(delay)、跳躍數)建立一控制拓撲，而傳送樹則由特定之資料傳送規則在此控制拓撲上建立。

在單資料來源之傳送樹回復的方法中，可分為 reactive[9]及 proactive[5-8]兩種。Reactive 方式為當成員偵測到 離開或失效後才開始重新建立傳送樹，而 proactive 方式則在離開或失效發生之前就預先計算備用路徑(backup path)，當偵測到成員離開訊息後即可立即利用備用路徑而進行傳送樹回復，因此能夠縮短傳送樹回復所需之時間。

3. Proposed Multi-source ALM and Recovery Approach

本論文提出一個新的應用層群播方法。首先，每一個群組成員皆會保存一個 Neighbor Information Table (NIT)。如圖一(a)所示，一個



圖一 Join procedure

NIT 之記錄包含了 neighbor_addr 及 hop_count 兩項記錄欄位。neighbor_addr 記錄此成員之鄰居成員之 IP 位置，hop_count 則為與此鄰居成員跳躍數。

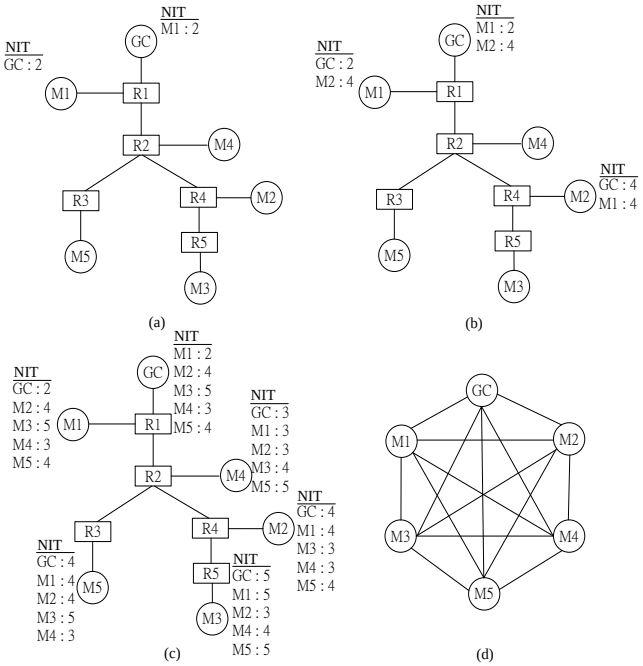
本論文在以下之描述中以「傳送(transmit)」來描述資料封包由原始資料來源端傳送至 next hop 的行為，「轉送(forward)」則為資料封包由中間節點轉送至 next hop 之行為，而「parent set」則為同一群組成員在各個不同來源端所形成之傳送樹上的 parent 的集合。

3.1 Multicast Group Management

此小節將描述應用層群播之群組管理之相關行為，包含了群組成員之加入(*Join)*及離開(*Leave)*等運作方式。

3.1.1 Member Join Operation

如圖一所示，欲加入群組之節點必須藉由任一種已存在之定址方法取得群組建立者之資訊，如文獻[17]所提出之 JXTA 之定址方法，並在取得群組建立者之資訊後，發送 Join_Req 給群組建立者，而群組建立者負責接受所有加入請求並發送 Join_Resp 至請求節點，而此 Join_Resp 將加入(Piggyback)目前群組中之所有成員的資訊。加入請求節點在接收此 Join_Resp 後將利用此成員資訊逐一發送 Probe 訊息至所有成員，當成員接收到此 Probe 訊息後，也將發送

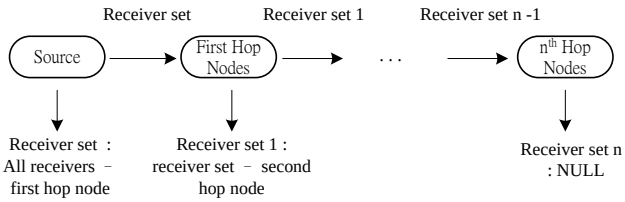


圖二 Example of member join procedure

Probe_Resp 至請求者，當加入請求者接收到所有成員之 Probe_Resp 訊息後，請求者即加入群組中，而所有成員在此一訊息交換的過程中也將會得到目前群組中所有成員之位址及其與自己之跳躍數，並且將此資訊記錄於 NIT 中。

圖二為成員加入群組之例子。此例中，GC (Group Creator)為群組建立者， R1 至 R4 為路由器，M1 至 M5 為群組成員並依此順序加入群組。如圖二(a)所示，當新成員 M1 加入後，原本已存在之成員 GC 將建立自己與新加入成員 M1 之記錄，而 M1 也依據 GC 之 Join_Resp 而建立與 GC 之記錄於 NIT。圖二(b)則為 M2 加入之過程，除了 M2 與 GC 互相得到彼此之資訊外，利用 GC 所回應之成員資訊 M2 與 M1 也能夠利用 Probe 與 Probe_Resp 訊息而建立彼此間之記錄。圖二(c)所示為當所有群組成員皆加入群組後，各成員所建立之 NIT 之狀態，圖二(d)則為所有成員加入群組後在應用層上所形成之完全網狀拓撲。

3.1.2 Member Leave Operation



圖三 Concept of transmit/forward method

當成員欲離開目前之群組之前，將發送 LEV 訊息給群組中之所有成員，使得其它成員能夠適時的將此成員從其 NIT 中移除。

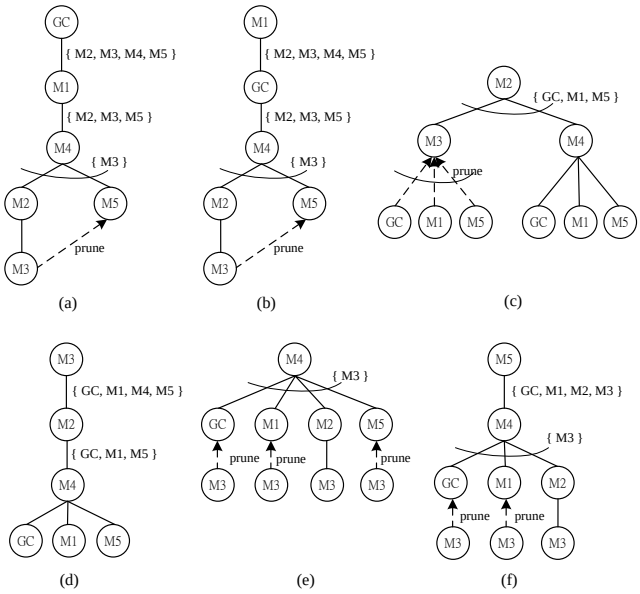
3.2 Data Delivery Tree Construction

本論文利用上述之群組管理所建立之 NIT，提出以下之 ***transmit/forward method*** (圖三)為群組成員在進行資料傳送或轉送時的準則，而此方法分散並同時在各群組成員間執行將為每一個來源端建立一個傳送樹。

3.2.1 Transmit/forward method

圖三所示為此 *transmit/forward method* 的主要概念。當任何一個來源節點欲傳送群播資料時，將在 NIT 中選擇與自己跳躍數最小之成員為傳送之下一轉送點，並且將群組中所有剩餘尚未在此次傳送之接收端以 receiver set 的方式加入至欲傳送之資料上。當任何一個成員接收到任一群播來源之群播資料後，若此資料中之 receiver set 不為空，將從此 receiver set 中選擇與自己跳躍數最小之接收端為傳送之下一轉送點，並將此次選擇之下一轉送點從此封包中之 receiver set 中移除再將資料傳送至下一轉送點。當成員接收到 receiver set 為空之資料封包時，因為 receiver set 為空表示此資料之所有接收端已有成員負責傳送或轉送，所以將停止此接收、選擇下一轉送點、更新 receiver set 及傳送的動作。

為了減少不必要的頻寬浪費，當成員接收由不同成員轉送之重複資料(duplicate data)，接



圖四 Source-specific delivery tree

收到重覆資料之成員節點將在這些重覆之轉送節點中選擇與自己之跳躍數較大之節點並發送 Prune 訊息至此節點，而接收到 Prune 訊息之轉送節點將停止轉送資料至此成員。

圖四(a)至(f)分別為以 GC、M1 至 M5 為來源端以上述之 *transmit/forward method* 所建立之傳送樹，實線為 *transmit/forward method* 所建立之傳送路路徑，虛線則為因重覆接收而被 Prune 訊息停止(disable)的傳送路徑。以 GC 之傳送樹為例(圖四(a))，當 GC 欲傳送群播資料時，依照其 NIT 所示(圖二(c))，M1 與其之跳躍數為最小，因此，GC 將選擇 M1 為傳送之下一轉送點，並將 receiver set 設定為{All group members}–{GC, M1} = {M2, M3, M4, M5}。當 M1 接收到此資料時也利用相同方式選擇 M4 為其轉送之下一轉送點，並將 receiver set 設定為{M2, M3, M4, M5}-{M4} = {M2, M3, M5}。利用相同的方式 M4 將選擇 M2 及 M5 為其轉送之下一轉送點，再設定 receiver set 為{M2, M3, M5}-{M2, M5} = {M3}。最後，M2 及 M5 將同時選擇 M3 為轉送之下一轉送點，當 M3 接收到來自 M2 及 M5 之重複之資料時，利用其 NIT，M3 可知 M2 與自

Session	Previous Hop (Parent)	Next Hop (Children)
Source 1		
Source 2	.....	
Source n		

(a)

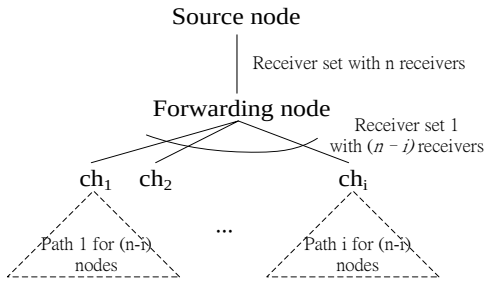
Session	Previous_Hop (Parent)	Next_Hop (Children)
M1	M1	M4
M2	M4	Leaf
M3	M4	Leaf
M4	M4	Leaf
M5	M4	Leaf

(b)

Session	Previous_Hop (Parent)	Next_Hop (Children)
GC	M1	M2, M5
M1	GC	M2, M5
M2	M2	GC, M1, M5
M3	M2	GC, M1, M5
M5	M5	GC, M1, M5

(c)

圖五 Forwarding Table (FT)



圖六 Backup routes maintained by transmit/forward method

己之跳躍數較小，因此，M3 將傳送 Prune 訊息給 M5，以停止 M5 繼續傳送重覆之資料。利用相同之方式，圖四(b)至(f)為以 M1 至 M5 為來源端之傳送樹。

由上述之 *transmit/forward method* 及例子可知，以此方式執行群播時，每一個群播成員皆必須建立一個如圖五(a)之 Forwarding Table(FT)，而此 FT 中的每個項目記錄著此成員節點對於各個來源端(session 欄位)之資料的轉送方式、上一轉送成員(previous hop)及下一轉送目的(next hop)等資訊，而當成員發生離開或失效時，則必需更新 FT，以避免影響資料的轉送。圖五(b)(c)則為相對於圖四中，GC 及 M4 之 FT。

3.2.2 Characteristics of Transmit/forward Method

Transmit/forward method 在群組成員之間的運作具有以下兩項特性使其能夠達到多資料來源群播及快速傳送樹回復的目的。

3.2.2.1 Establishing Per-source Tree Structure for Multi-source Multicast

每一個接收到資料之成員根據資料的 receiver set 而決定此資料所必須轉送的下一轉送點的方式使得成員在所形成的各個傳送樹中的 parent-child 關係將隨著來源端附帶在資料中不同的 receiver set 而不同。如圖五所示，成員之 FT 所建立之 parent 欄位對於不同來源端所建立的轉送規則有不同之 previous hop (parent)節點，而這些 parent 的集合為此成員對於所有來源端之 parent set，如圖五所示，在此例中，GC 及 M1 至 M5 之 parent set 分別為{M1, M4}、{GC, M4}、{M3, M4}、{M2}、{GC, M1, M2, M5}及{M4}。

任一成員對於任一來源端所建立的轉送規則(圖五)表示了此成員對於此來源端資料的轉送方式，也表示了此成員在以此來源端為頂點而形成的傳送樹的部份結構，因此，所有參與群播之群組成員對於某一來源端所建立之轉送規則即可完整的建構出此來源端之傳送樹。換句話說，各個傳送樹的建立及維持是藉由所有成員的分散式執行的，而不需由特定成員執行，因此既可以為每一個來源端建立一傳送樹又可避免特定成員之負載及單點失效等問題。

3.2.2.2 Providing Backup Routes for Proactive Recovery

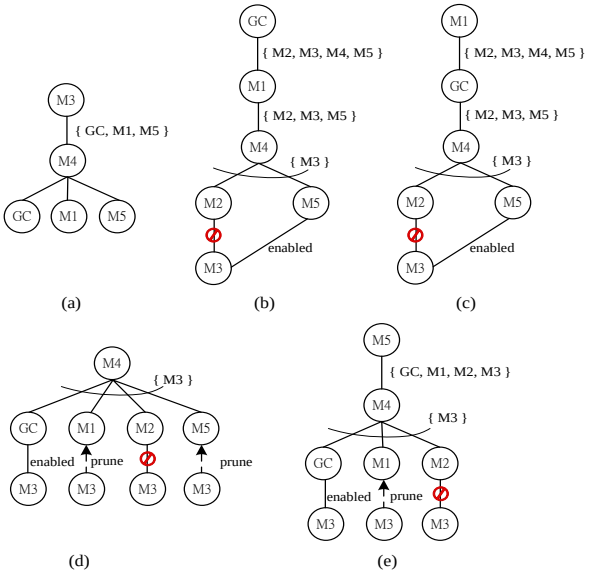
Transmit/forward method 所建立之傳送樹中，傳送樹之分支都使用相同之 receiver set，如圖六所示，當 forwarding node 從 source node 接收包含有 n 個 receiver 的 receiver set 之群播資料時，根據 *transmit/forward method*，forwarding node 將從此 n 個 receiver 中挑選下一轉送點，假設若是 forwarding node 從此 n 個 receiver 中挑選 *i* 個

下一轉送點(假設若為ch_1 - ch_i)，並將此 *i* 個下一轉送點從 receiver set 中移除而形成 receiver set 1。此時，(ch_1 - ch_i)所接收資料之 receiver set 皆為相同之 receiver set 1。藉由 transmit/forward method 的傳送方式，receiver set 1 中之 receiver 能夠透過(ch_1 - ch_i)所形成之任何一個子樹而接收到 source node 的資料。換句話說，(ch_1 - ch_i)將為 receiver set 1 中之 receiver 建立 *i-1* 條備用路徑。

3.3 Proposed Proactive Recovery Approach

群組成員離開群組的原因可分成以正常離開程序離開及因此失效而離開兩種。當成員以正常程序離開群組時，將會先傳送 LEV 訊息給所有群組成員，而當成員因為發生失效而離開群組時，其它群組成員將無法得知此訊息，因此，在本論文之所有群組成員將利用週期性的發送 *Heartbeat(HB)*訊息給其 parent set，藉由此訊息，所有傳送樹之成員即可得其所有下一轉送點的是否還存在於網路中。

本論文所提出之傳送樹回復方法為讓離開成員之 parent 在以 LEV 或 HB 訊息而偵測到離開訊息後，使用 *transmit/forward method* 所產生之備用路徑來為受到影響之成員重新建立路徑。當成員 *i* 傳送 LEV 訊息或因 failure 而停止傳送 HB 訊息時，其 parent set 中之 parent 皆會偵測到此成員 *i* 已離開群組，此時，所有 parent 將會把成員 i 從其 NIT 及 FT 之所有轉送規則中包含有成員 i 之 next_hop 記錄中移除。而當成員 i 從 next_hop 中移除後，可能會造成轉送規則中之 next_hop 為空之情況，當 next_hop 為空時則表示以此 parent 為頂點之子樹中的所有成員將受到離開成員的影響而中斷接收資料(因已無任何 next hop 可進行轉送工作)。此時，parent 將從 NIT 重新選擇下一轉送點並重新設定 receiver set，再將含有新的 receiver set 的資料傳送至新的下一轉送點。另一種情況，若是移除成員 i 後之 next_hop 不為空時，表示受影響之成員為以離開成員為頂點之子樹(此子樹之大小必定小於 next hop 為空時之子樹大小)，但在此情況中，根據前述之 Backup Route 之特點，所有在 next_hop 中之其它成員所形成之子樹皆能夠提供這些受影響之成員另一條備用路徑，因此，傳送樹回復的方式則只要此 parent 重新傳送 receiver set 給所有未離開之下一轉送點以啟用備用路徑即可。在前述之 next_hop 為空或不為空時的傳送樹回復方式，parent 所傳送之 receiver set 都包含了所有因成員離開而受影響的成員，因此，受影響之成員最後都將會被重新選為下一轉送點而重新回至傳送樹上。由上之敘述中可看出，parent 在重新建立傳送樹的過程中只需進行查表動作 (look-up 及 remove)而決定下一轉送點及 receiver set 即可，也因此可以大量的減少傳送樹回復所需的時間。



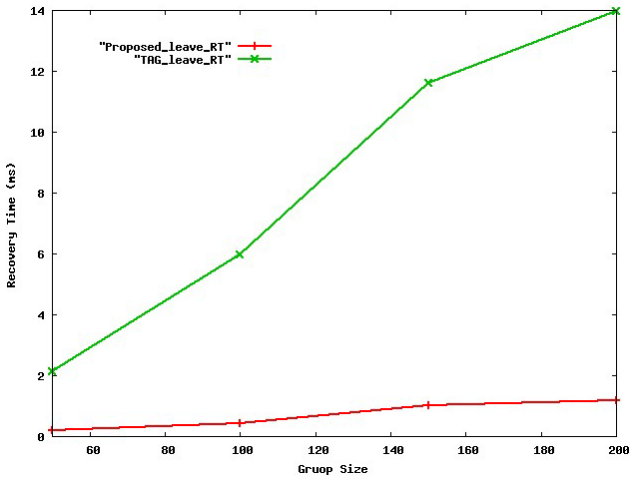
圖七 Path recovery

圖七為接續圖四建立傳送樹後之一傳送樹回復之例子。此例子假設 M2 離開群組後之各傳送樹的回復過程。當 M2 離開群組後，其 parent set 中之成員 {M3, M4}將分別把 M2 從其 NIT 及各來源端之 next_hop 記錄中移除。由圖四(d)可知，在移除 M2 後，M3 之 next_hop 將為空，

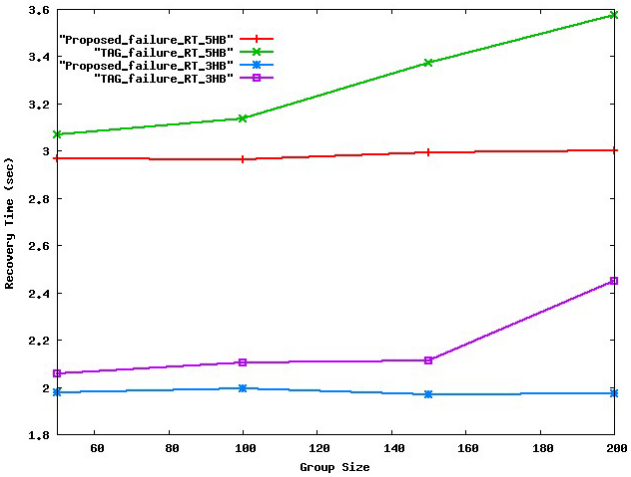
因此，M3 將重新選擇 M4 為下一轉送點，並設定 receiver set 為{GC, M1, M5}，而 M4 接收資料後也將根據 receiver set 重新選擇下一轉送點為 GC, M1 及 M5，如圖七(a)所示，受影響之成員{GC, M1, M4, M5}都將重新回到傳送樹上。另一方面，當 M4 利用相同方式移除 M2 後，由圖四(a)(b)(e)(f)對照可知，M4 在各個傳送樹中之下一轉送點在移除 M2 後，仍然有其它轉送點可繼續傳送資料，因此，M4 只需要重新傳送 receiver set 給未離開之轉送點，並執行 *transmit/forward method*，因為此 receiver set 以包含了所有受影響之成員，因此，受影響之成員將在執行 *transmit/forward method* 後而回復至傳送樹中，圖七(b)(c)(d)(e)分別為 M4 recover 後之 GC、M1、M4 及 M5 的傳送樹。

4. Performance Evaluation

目前對於應用層群播的研究中，還未有任何的研究提出具有多資料來源及快速傳送樹回復能力的應用層群播。因此，本論文將以 single share tree 的方式實作於單資料來源之應用層群播，使其具有多資料來源群播之能力。在單資料來源應用層群播的研究中[1-4]，文獻[2]使用拓撲資訊為傳送樹之建立依據而有較好之效能。因此，我們將與文獻[2]所提出之 TAG 在傳送樹



圖八 Average recovery time for nodes leave



圖九 Average recovery time for nodes failure with various Heartbeat frequencies

回復時間(recovery time)、控制訊息耗費(control overhead)及資料傳送延遲(data delivery delay)等三項效能指標進行效能比較。

4.1 Simulation Setup

本論文模擬所採用之拓撲模型(topology model)為 GT-ITM[18]所產生之 transit-stub 拓撲模型，拓撲大小為 492 routers，連結之 latency 為 10-100ms，群播群組大小為 50-200 並且以 unifrom 之分佈連接至拓撲模型上之 stub nodes，每一個群組成員成為來源端之機率為 0.1。模擬時間為 3600 秒，成員之 leaving 及 failure 行為彼此獨立，並假設為速率 λ = 4/minute 的卜松分配(Poisson)，來源端傳送資料之速率分別為 128kbps 及 256kbps 兩種速率。

4.2 Simulation Results and Discussion

4.2.1 Recovery Time(RT)

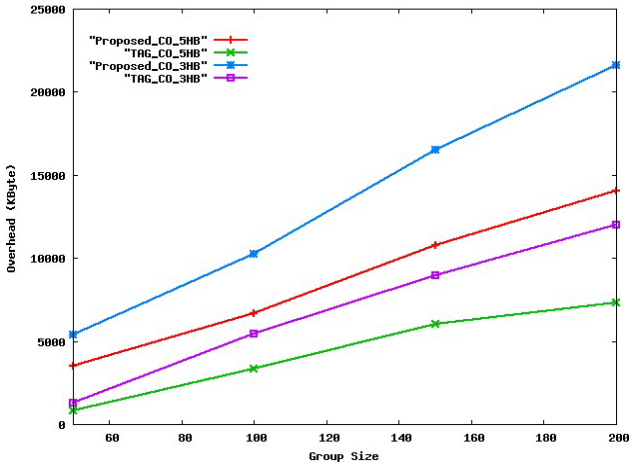
圖八為成員 leave 後，傳送樹之回復時間。如 3.3 節所提，本論文所提出之 *transmit/forward method* 所產生之備用路徑使得回復的時間只需要由離開成員之 parent 進行查表動作，因此大幅的縮短回復時間且也較不會隨著群組大小變大而上升。而 TAG 之回復方式，因成員需要重

新加入群組，所以隨著群組大小的增大，其重新加入群組所需之訊息交換及等待回覆所花費之時間也會隨著群組大小增大而上升。

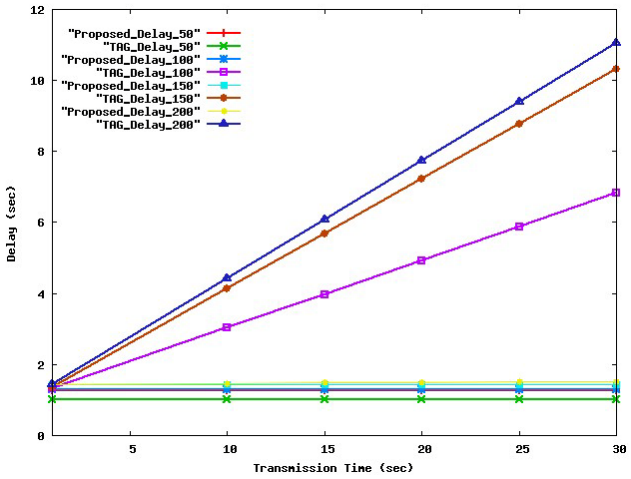
圖九為成員 failure 後，傳送樹之回復時間，如圖所示，若 HB period 愈小則兩種方法皆能愈早偵測 failure 的發生而進行回復因此也能縮短回復時間。本論文之 failure 回復時間在 HB period 為 3 秒及 5 秒時均低於 TAG 之 failure 回復時間。

4.2.2 Control Overhead (CO)

控制訊息耗費為進行傳送樹回復而產生之網路流量。TAG 之控制訊息耗費主要來自當 leave 或 failure 發生後，成員重新加入群組時所產生之訊息及 HB 訊息，而本論文之控制訊息耗費則主要來自於 HB 及 Prune 訊息。如圖十所示，為了避免所有來源端之資料皆由同一傳送樹傳送而造成 SPOF 之問題，本論文所提出之 *transmit/forward method* 使得每一個成員必須向其 parent set 發送 HB 訊息，因此，會有較大之控制訊息耗費。然而，在模擬過程中，所有來源端所產生之群播資料與控制資料之總合在不同之群組大小時約為 579 Mbytes 至 2319 Mbytes 之間，然而，本論文之控制資料約為總資料量之 0.6%。因此，為了達到多資料來源群播的目



圖十 Average control overhead



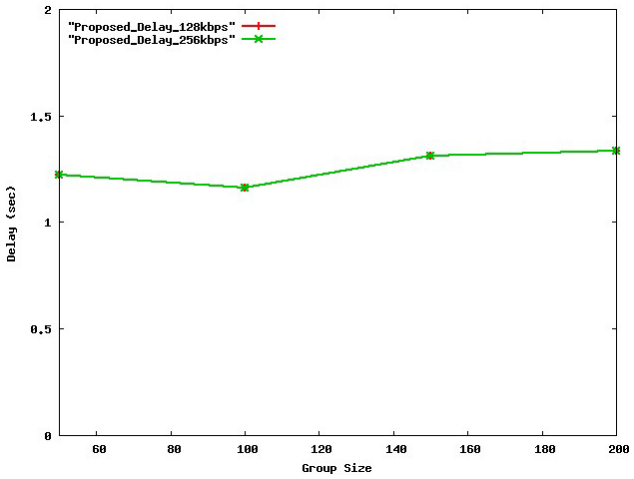
圖十一 Average delay

的且能避免傳輸瓶頸問題的要求下，此控制資料量相對於總資料量是足夠低且能夠被接受的。

另一方面，從圖九及圖十可以看出，failure 的回復時間與控制訊息耗費有相當大的關係。當 HB period 較短時，表示 HB 訊息傳送的頻率上升，因此，成員之 failure 也能夠較快的被偵測出而縮短回復的時間，但同時控制訊息耗費也會因 HB 訊息傳送次數的增加而增加。反之，若是 HB period 較長，HB 訊息傳送的頻率則較不頻繁，控制訊息耗費下降，但回復之時間則相對的上升。因此，控制訊息耗費與 failure 之回復時間為一交易(tradeoff)之問題，可視實際之應用而做調整。

4.2.3 Data Delivery Delay

在實際的群播應用中，並非所有之成員皆會接收所有來源端之資料，換句話說，任一來源端之群播對象並非為此群組中的所有成員。因此，本論文在資料傳送延遲的模擬上將分為無來源端選擇(source selection, SS)及有來源端選擇兩種模擬環境。前者表示，任一個來源端將群播其資料至所有成員，而後者則表示每一個成員成為任一來源端之接收端的機率為 p。



圖十二 Average delay with source selection

圖十一所示為無來源端選擇之延遲比較。TAG 利用拓撲資訊來輔助建立傳送樹的方式雖然能夠較準確的依據成員之間在網路拓撲上的連接關係而建立延遲效能較佳之傳送樹，但由於以 single shared tree 為多資料來源群播之實作方式，因此，所有群組中之來源端皆透過傳送樹之頂點將資料傳送至所有接收端，因此，當群組大小或來源端個數增加時，頂點即成為傳送之一傳輸瓶頸。如圖十一所示，當群組大小為 50 時，TAG 之 delay 仍然能夠有較好之效能，然而，隨著群組大小及來源端個數的增加，延遲也快速的上升。

在有來源端選擇之模擬中，成員對任一個來源端之選擇接收之機率 p 設定為 0.25。如圖十二中，本論文之多資料來源應用層群播架構之資料傳送延遲在兩種傳輸速率下皆能夠穩定(stabilize)，這也表示本論文所提出之方法能夠適用於此兩種傳輸速率之多資料來源群播。

5. Conclusion

本論文針對了多資料來源之應用而提出了一多資料來源應用層群播及其傳送樹回復的方法。在所提出的方法中，*transmit/forward method* 為每一個來源端建立一個傳送樹，因此，能夠避免 single share tree 所可能發生之 SPOF 問題，而 transmit/forward method 之 backup route 之特性，使得 leave 或 failure 之回復只需進行查表動作，而縮短了回復的時間。透過模擬驗證了本論文所提出之方法雖然有較文獻[2]高之控制訊息耗費，但是對於為了處理多資料來源群播所產生的大資料量來說，仍然是可接受的。因此，本論文所提出之方法在多資料來源群播的應用上，能夠有較短之回復時間外，且也能避 single share tree 之 SPOF 及傳輸瓶頸等問題，且也有較小之資料傳送延遲。

6. Reference

[1] Y.H. Chu, S. G. Rao, S. Seshan, and H. Zhang, "A Case for End System Multicast," IEEE J. Selected Areas in Comm., vol. 20, no. 8, pp. 1456-1471, Oct. 2002.

[2] M. Kwon, and S. Fahmy, "Topology-Aware Overlay Networks for Group Communication," Proc. of 12th international workshop on network and operation systems support for digital audio and video. pp. 107-136, May 2002.

[3] Mojtaba Hosseini, Dewan Tanvir Ahmed, Shervin Shirmohammadi, and Micolas D. Georganas," A Survey of application-layer multicast protocols, "IEEE Communications Surveys, volume 9, No. 3, 3rd Quarter, pp. 58-74, 2007.

[4] L. Baiqiang, T. Takeshi, and K. Keiichi, "Honeycomb: A Novel Approach for Construction of Stable ALM Overlay," Fifth Internetional Conference on Info. Tech.: New Generations, pp. 402-407, April 2008.

[5] M. Yang and Z. Fei, "A Proactive Approach to Reconstructing Overlay Multicast trees," Proc. INFOCOM '04, Mar. pp. 2743-2753, 2004.

[6] T. Kusumoto, Y. Kunichika, J. Katto, and S. Okubo, "Proactive Route Maintenance and Overhead Reduction for Application Layer Multicast," Proc. ICAS-ICNS'05, pp. 17-17, Oct. 2005.

[7] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan, "Resilience Multicast Using Overlays," Proceeding of ACMSIGMETRICS' 03, pp.102-113, June 2003.

[8] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan, "Resilience Multicast Using

Overlays," IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 14, NO. 2, pp. 237-248, April 2006.
[9] H. Deshpande, M. Bawa, H. Garcia-Molina, "Streaming Live Media over Peers," Technical Report 2002-21, Standfor Univ., Mar. 2002.
[10] G. -In Kwon and J. W. Byers, "ROMA: Reliable Overlay Multicast with loosely Coupled TCP Connections," Proc. INFOCOM ’04, Mar. p. 395, 2004.
[11] F. Baccelli, A. Chaintreau, Z. Liu, A. Riabov, and S. Sahu, "Scalability of Reliable Group Communication Using Overlays," Proc. INFOCOM ’04, p. 430, Mar. 2004.
[12] Y.H. Chu, S. G. Rao, S. Seshan, and H. Zhang, "Enabling Conferencing Applications on the Internet using an Overlay multicast Architecture," ACM SIGCOMM, pp.55-67, Aug. 2001.
[13] B. Zhang, S. Jamin, and L. Zhang. "Host Multicast: A framework for delivering multicast to end users," Proceedings of IEEE INFOCOM, pp.1366-1375, June 2002.
[14] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable Application Layer Multicast," ACM SIGCOMM, Pittsburgh, pp. 205-217, Aug. 2002.
[15] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Application-Level Multicast Using Content-Addressable Networks," Proc. Int’l Workshop networked Group Comm. (NGC’01), pp.14-29, 2001.
[16] R. Zhang and Y.C. Hu, "Borg: A Hybrid Protocol for Scalable Application-Level Multicast in Peer-to-Peer Networks," Proc. Int’l Workshop Network and Operating System Support for Digital Audio and Video (NOSSDAV), pp.172-179, 2003.
[17] Sun Microsystem, "JXTA v2.0 Protocols Specification," http://www.jxta.org
[18] E. W. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," Proceedings of INFOCOM’96, pp. 594-602, san Francisco, CA. 2002.