效率型網路群播協定之研析

Design and Implementation of Efficient Network Multicast Protocol

呂崇富
中國海商專校 資訊工程科
peter@mail.ccmtc.edu.tw

摘要

傳統伺服器傳輸方式為伺服器將資料用星狀的架構傳送給所有需要下載的使用者，因此伺服器上傳頻寬會被不斷切割耗盡，伺服器因身為資料提供者，所以其下載頻寬總是空閒而無法利用，但同一時間對於使用者來說，卻也只用到下載的頻寬而已，使用者之上傳頻寬仍是空閒而未加利用，因此對於多人分享同一份資料而言，並沒有充分地利用到使用者的所有頻寬。本篇文章研究目的在於探討如何在 UNICAST 網路架構下有效利用每個網路節點上傳與下載頻寬，讓一份資料的群播更有效率。我們提出的網路群播協定，就是反應了寬頻網路時代下對等式(P2P)傳輸的需求，除了透過伺服器來媒介使用者與使用者間的傳輸以外，更同時利用每個使用者的上傳與下載頻寬，在某個使用者下載一份資料之餘，更可同時將這份下載中的資料分享給更多人。

關鍵詞：頻寬、伺服器、客戶端、節點品質因素、路由表。

一、簡介

在網際網路應用上，伺服器 (Server)一直扮演很重要角色，一方面整合資訊，另一方面也提供資料給各個客戶端 (Client)，以往伺服器總是將自己的對外頻寬分享給每一個客戶端，例如 FTP Server 提供下載與上傳檔案服務，讓檔案的分享更加容易。隨著家用寬頻網路普及與多媒體資料下載風氣盛行，以往伺服器已不太能負荷現今動輒至少 512K 下載頻寬以及動輒數十 MB 檔案傳輸之流量需求，網際網路的骨幹頻寬沒有大幅成長，但是使用者的頻寬卻成長 10 倍至 30 倍不等，原有的伺服器架構已經無法負荷新的寬頻使用者族群。

如果要以技術層面來提昇網際網路骨幹之頻寬況日費時，且花費成本和時間更是無法估計。直到約 1999 年興起的 P2P(Peer-to-Peer)技術問世[1,6]，才稍微分擔與解決部分伺服器的負擔，P2P 技術特色是能夠讓使用者「直接相互分享」，打破了傳統使用者僅可擔任客戶端之觀念，讓檔案分享更加便利與普遍，而原來的伺服器只需做到媒介傳輸的角色，伺服器本身並不用分擔實際傳輸檔案之頻寬。我們提出的網路群播協定，正反應了寬頻網路時代下 P2P 傳輸理論的精神並加以演化，除了透過伺服器來媒介使用者間的傳輸外，更同時利用每個使用者的上傳與下載頻寬，當某個使用者下載一份資料之餘，更可同時將這份下載中的資料分享給更多人。

本網路群播協定設計理念為在寬頻環境中，同一時間能充分利用伺服器(由某個不特定之檔案分享使用者擔任)及其客戶端的所有上傳及下載頻寬，以加速資料分享之效率。伺服器每次丟一個封包給其下層之特定客戶端，這個客戶端拿到封包後馬上就利用其尚未使用的上傳頻寬傳送給其他客戶端，而其他客戶端收到封包後則依照此概念反覆操作，如此一來伺服器只需要提供資料給少數客戶端，而這些客戶端會自動把資料分層的分享下去。

在這樣的觀念革命下，我們不再使用特定伺服器這樣的名稱與角色，也沒有所謂單純的客戶端，因每個客戶端都可能充當伺服器把資料分享給其他節點，取而代之是主控端和發散節點的概念。以本協定為藍本的檔案分享系統，使用者可建立自己的檔案分享社群，能讓一群好友快速便利的分享彼此擁有的檔案，而所分享之檔案型態並沒有特定檔案結構之限制，舉凡純文字、影音多媒體…等均適用。

二、效率型網路群播協定

理論上，全雙工的鏈狀傳輸比主從式星狀的頻寬效益好[4,5]，但在實際傳輸中完全的鏈狀會碰到很多問題，其中最大問題是因每個節點頻寬並不相同，而鏈狀架構的完美預設是每個節點的對外頻寬與傳送速度皆相同，但很顯然事實並非如此，且因為是合作性質的依次轉送封包，所以每個節點收到封包都會有累積性的延遲，而鏈狀結構一直延伸到大量(如數千個)節點的時候會發生更多問題。

以上所探討的問題在主從式的星狀結構並不嚴重，但星狀最致命的缺點就是要付出頻寬代價，所以本協定的演算法將同時採用了星狀與鏈狀架構加以整合，成為雙軌運行架構。本協定演算法的原則是以星狀架構為優先，每個發散節點會嘗試以主從式檔案分享方式同時分享給多個節點，直到其上傳頻寬剩下不足 1/5 為止，也就是說每一個發散節點(資料提供或是轉送者)會先嘗試充分利用頻寬同時送給多個下游節點，而如果頻寬不足以送給多個下游節點的情況下，就會以鏈狀架構只提供一個下游節點。也就是說，路由網路要採取環狀或是星狀全憑該節點所擁有的上傳頻寬而定，唯獨鏈狀傳輸如果連續到第五層的情況下，會強迫改採星狀傳輸以防止網路的癱瘓和延滯。

在本協定中並沒有明顯的特定伺服器(Server)和客戶端(Client)，只有分為主控端、發散節點和末端節點。也就是說每個網路節點都有可能擔任 Client 和 Server 的功能，整個網路傳輸架構圖則有點類似樹狀結構呈發散狀成長，而以星狀傳輸和鏈狀傳輸並用的方式將資料傳輸至每個節點。

本協定最大目的為突破以往 Server/Client 的資料分享與傳輸模式，為此我們使用鏈狀連結與星狀發散並用的方式，將資料接收端分成 Tier 1、Tier 2、Tier 3 …的群組，當原始資料發送端要傳送資料給所有人的時候，只需要先傳送給 Tier 1 的節點群，而 Tier 1 節點群會自動擔負起下層資料傳送的功能，並依序將各個資料傳送給 Tier 2，而 Tier 2 也會以相同的方法持續把資料一層一層的往下送，如圖 2.1 所示。如此一來，除了 Tier N 的末端節點以外，每一層的節點事實上都擔負了 Server 的功能，而且自己本身也是 Client，就像水系分流一樣，由主幹先流到支幹，再由支幹流到末端，且每個 Tier Group 間資料傳輸間隔只有一個封包的時間差距，因此整個架構中的所有節點幾乎都可以同一時間收到一筆資料。

本協定主要以兩個演算機制為核心，分別為搜尋節點與測速演算機制及路由演算機制。有關本協定之整體機制架構圖及演算法流程圖，分別詳如圖 2.2 及圖 2.3 所示。

(一) 搜尋節點與測速演算機制

路由演算機制為掌握傳輸效率的最大關鍵，為了更有效率的建立路由網路，必須要將每個節點加以分類層級，也就是要去評斷一個節點的傳輸品質好壞並作出排名，最後再取品質最好的節點當作是傳輸主控端，負責整個網路的路由演算和路由表分配、錯誤修正等。事實上只有資料提供者，才具有主控端節點的資格，若不是資料提供者，即使作為主控端節點並沒有好處，因為擁有原始資料的節點，仍然要將資料傳送給主控端節點，再由主控端節點發送給下游節點，因此主控端節點以原始資料提供節點擔任為宜，然後以頻寬測速的方式算出節點品質因素(NQ)，續而排列節點的傳輸品質優劣。有關每個節點建立其本地端連線速率表、整合連線速率表並計算節點品質因素(NQ)之演算流程，分別詳如圖 2.4 及圖 2.5 所示。

(二) 路由演算機制

不好的路由表會將整個傳輸效率拖慢到幾乎不可置信的地步[8,9]，傳輸效率的關鍵點在於如何演算出正確的路由表，先前演算出主控端與節點品質因素也是為了這個原因。路由表建立演算機制為星狀與鏈狀並用，至於何時該採取哪一種路由方式則全憑協定演算而得，有關路由演算法之流程詳如圖 2.6。

路由表建立演算機制是預先以測試數值為基礎來運算出理想路由表，但在實際傳輸情況卻會有所改變，例如某一負責多個子節點的發散節點突然變得十分緩慢，或是某節點突然斷線…，為了解決這些問題，需要一個路由表修正機制來維持協定的穩定性與效率性。路由表修正方式則端看節點數量多寡而定，若總節點數不大，則重算路由表，這部分與圖 2.6 的路由表建立演算機制相同；若總節點數龐大，則須先清除要分配節點或是節點群的連結標記以及其從屬關係之設定後，再以區域演算的方式修正路由表，而此類路由表修正演算方式可分為增加節點、刪除節點、慢速節點處理、主控端斷線處理等四種狀況，其路由表修正演算法之流程分別詳如圖 2.7 至圖 2.10 所示。

(三) 封包及資料同步、加密機制

1. 封包規劃：傳輸封包欄位規劃如表 2.1。
2. 封包同步：

當一個新的節點加入或是因故須大幅重建路由表時，都有可能會發生資料傳輸不同步的問題，當遇到這些狀況時，則會由子節點和父節點互相確認封包編號。

(1) 如果父節點的封包編號較大，則會將尚未傳輸的封包指派給子節點。

(2) 如果子節點的封包編號大於父節點的情況，父節點會依照路由層級向更上層的父節點要求封包，並且暫停對於該子節點的傳輸直到封包完全同步為止。

(3) 如果子節點的封包編號大於父節點，但因層級要求封包至最上層主控端仍無此封包的情形下，則放棄傳輸。

3. 傳輸偵錯：

TCP/IP 在傳輸層即有錯誤偵錯功能[11]，所以並不須特別提供錯誤偵錯的機制。

4. 資料的加密與解密：

在封包規劃中預留一個加密欄位，提供將來放置加密資訊的擴充需求，基本上協定層的加密是採取對稱式加密，其餘指令欄位、路由表欄位及資料欄位等，經由這把金鑰的運算後可以回復至協定可判讀的資訊。

三、演算法評估與實作

在現實網路中尋找大量節點來實測演算法有效性並不切實際，且區域網路中須使用 Switch Hub 才能達到每個網路節點中全雙工傳輸的效果，所以大量節點路由網路圖的建立、修正及傳輸的測試部分，我們是以一個模擬環境來測試。利用 Visual Basic .NET [2,3]實際開發了一個路由表建立、演算、修正以及實際傳輸的模擬環境[7,10]。

(一) 固定所有節點並使用單一外在因素

在第一個測試模擬實驗裡面，我們先簡化外在條件，假設所有節點皆具同樣的對等上下傳頻寬(64Kbps)、全雙工傳輸模式以及其他相同條件因素下，模擬計算將一筆 1024 Kbpye 資料傳送給 20 個節點的所需時間。在這個測試環境中，因為所有節點的條件都相同，所以經由模擬系統運算出來的 NQ 值皆為 64，在本協定的設計裡會依照 NQ 值大小之排列來決定主控端節點(Control Node)，而因為所有節點的 NQ 值都為一樣，所以會尋找陣列中最末端的節點當作是主控端點。

圖 3.1 為以本協定路由演算法機制演算出來的路由網路圖，為完全鏈狀的理想傳輸圖，因每個節點的頻寬都是一樣，於彼此都無能力再繼續供給其他節點的情況下就會採用鏈狀模式一直連結。在少數節點的情形下這種傳輸方法是可接受的，但於大量節點數時，其延遲時間將被累積，而產生難以置信的延滯。

由模擬結果得到，以本協定之路由演算機制所規劃出的網路，其傳送 1024 Kbyte 資料給所有節點的平均傳送時間是 137 秒。但在同一外在因素下的傳統主從式星狀之架構的網路，其平均傳送時間則需 2432 秒，是本協定演算法的 17.75 倍。

(二) 以亂數決定節點外在因素

我們進一步以亂數來決定每一節點之上傳、下載頻寬以及工作模式等數值，以求模擬結果能更接近於現實狀況，仍將模擬計算將一筆 1024 Kbpye 的資料傳送給 20 個節點的所需時間，而所有節點及頻寬等資訊詳如表 3.1。

以本協定路由演算法機制演算出來的路由網路圖詳如圖 3.2，從圖中可看出演算機制會自動判斷節點外在因素後，再決定將資料以星狀或鏈狀方式來傳輸。由模擬結果得到，以本協定之路由演算機制所規劃出的網路，其傳送 1024 Kbyte 資料給所有節點的平均傳送時間是 112 秒，而大部分節點在 150 秒以內就可完成接收所有資料 (除了少部分下載頻寬較慢的節點外)。但在同一外在因素下的傳統主從式星狀架構的網路，將由頻寬最大的 Server 端將資料同時傳送給所有節點，其平均傳送時間則須 518 秒，是本協定演算法的 4.625 倍。

詳如表 3.2 所示，在多次的模擬測試結果後，我們發現以傳統星狀傳輸需要的時間分布在 500~1000 秒間，而本協定路由演算法則分布在 200~300 秒間，因此本協定的演算效率約是傳統星狀網路的 3 倍上下，而這個數值會隨著路由演算法的改進而會有更大的成長空間。

(三) 實際傳輸測試

我們將選擇一個簡單測試環境，並撰寫一個測試平台，其核心使用本協定的 ActiveX Object 來測試路由演算法的建立機制、測速機制，以及實際資料傳輸的效益性及傳輸速度，並與傳統主從式架構作一系列的比較。

我們將測試用的程式平台安裝在十台電腦上執行，其中一台當作資料原始提供端(主控端)，而其他電腦皆為下游節點，一開始由主控端自動以 IP Scan 的方式掃描整個 Class C 的網路中是否有電腦執行了測試平台，在這邊

我們是以 Port 2005 作為掃描依據，如果該 IP 有監聽 Port 2005 則代表該 IP 的電腦執行了本協定的測試平台，這樣做的原因是可以省略一個 Server 的媒介，節點和節點之間在知道對方 IP 位置、Command Port、Data Port 的情況下，可以更快速及簡便的進行傳輸，但此測試方法不能在實際的 Internet 上使用，僅為在 Local LAN 測試時使用。在實際測試中，本協定之搜尋節點與測速演算機制僅花費了 260 毫秒即成功的搜尋到所有安裝執行了測試平台之電腦節點，並成功的運算出每個節點的平均頻寬及全/半雙工測定值，詳如表 3.3。

我們發現因區域網路速度很快，有時候作業系統所造成的延遲硬碟存取速度反而成為傳輸時的瓶頸，然在這個測試中得知本協定的效率大約是傳統主從式架構的 2~3 倍。

四、結論

效率型網路群播協定改善傳統主從式星狀網路架構的頻寬使用不平衡之狀況，改以環狀或樹狀方式來傳送/接收資料，可比原始的主從式架構提高 3~7 倍的傳輸效率，為解決現階段網路頻寬不足的一種低成本解決方案。

本協定雖以 P2P 檔案傳送應用來證實其效率與有效性，但協定本身可以應用於任何層面上，可將一份資料快速的傳播給所有人，也可以自動排序並分配欲連線到主控端的使用者。本協定的特點為在傳輸資料量越大且總資料節點越多的情況下，越能突顯其效率，舉例來說，如果某一台 Server 的目的為擺放文字型的聊天室，因為文字傳送的頻寬佔用很少，所以使用本協定就喪失其效益性，反而還會因為複雜的資料路徑與路由演算法而造成硬體與頻寬上的額外負擔。所以本協定的適用範圍以多媒體、非資料保全性(有可能會遺失封包)或大量性的資料傳送為佳，例如：P2P 模式的檔案分享；隨選視訊、即時性或是非即時性的網路廣播；多人語音與視訊會談；以多媒體資料為主體的網頁內容或是相關電子媒體…等。

誌謝

對於本人指導之專題學生全力協助本協定與模擬工作，在此一併致謝。

五、參考文獻

[1] 吳明蔚,林盈達, “P2P 打破網路藩籬”, 網路通訊, vol.132, pp. 53-61, 七月, 2002.

[2] 黃嘉輝, “Visual Basic 網際網路程式設計-TCP/IP 與 Internet”, 台北:文魁, 2000.

[3] 葛湘達, “Visual Basic 秘密提示”, 台北: 第三波, 1999.

[4] Cisco Systems Inc., Product Documentation, 2002, http://www.cisco.com/univercd/home/home.htm

[5] Cisco Systems Inc., Protocol Brief - TCP/IP, 2002, http://www.cisco.com/warp/public/cc/techno/protocol/iprout/tech/tcpip_pc.htm

[6] Clipcode Peer Team, P2P Protocol and Object Model, Working Draft 1st June 2001, http://www.clipcode.org/peer/p2p_protocol_req/20010601/

[7] Free Protocols Foundation, The Protocol Development Process, 2002, http://www.freeprotocols.org/freeProtocolProcess/split/node3.html

[8] Jim Carr, “Routing Protocols”, Network Magazine, 06/01/91, http://www.networkmagazine.com/article/NMG20000724S0041

[9] J. Sum, H. Shen, G.H. Young, J. Wu, C.-S. Leung, “Analysis on extended ant routing algorithms for network routing and management”, Proc. 2nd International Conference on Parallel and Distributed Computing Applications and Technologies, Taipei, 2001.

[10] National Cancer Institute, Guidelines and Tools for Protocol Development, 2002, http://ctep.cancer.gov/guidelines/

[11] RADCOM Equipment Inc., Protocol Directory - TCP/IP Reference Page, 2002, http://www.protocols.com/pbook/tcpip1.htm

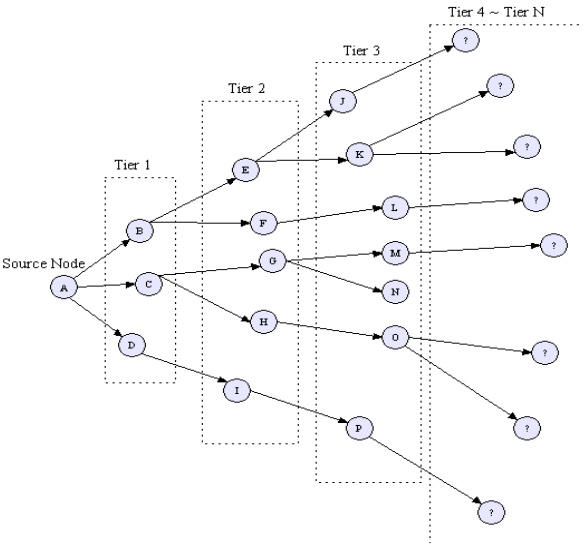


圖 2.1 協定運作示意圖

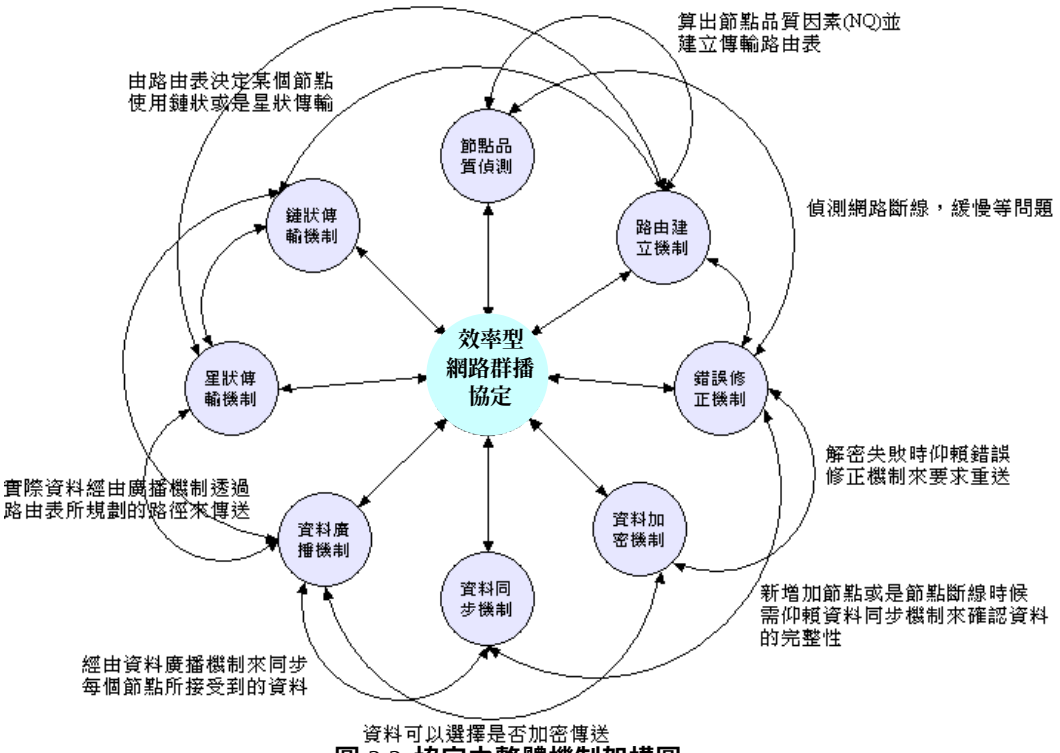


圖 2.2 協定之整體機制架構圖

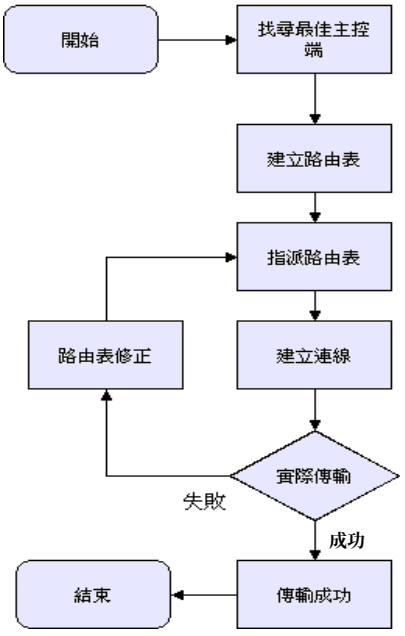


圖 2.3 協定之整體演算法流程圖

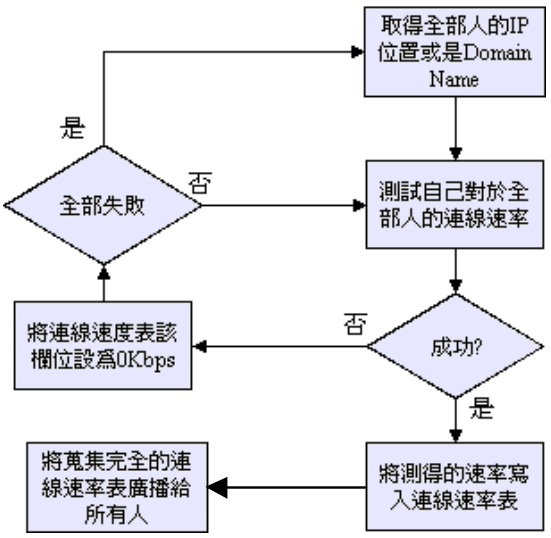
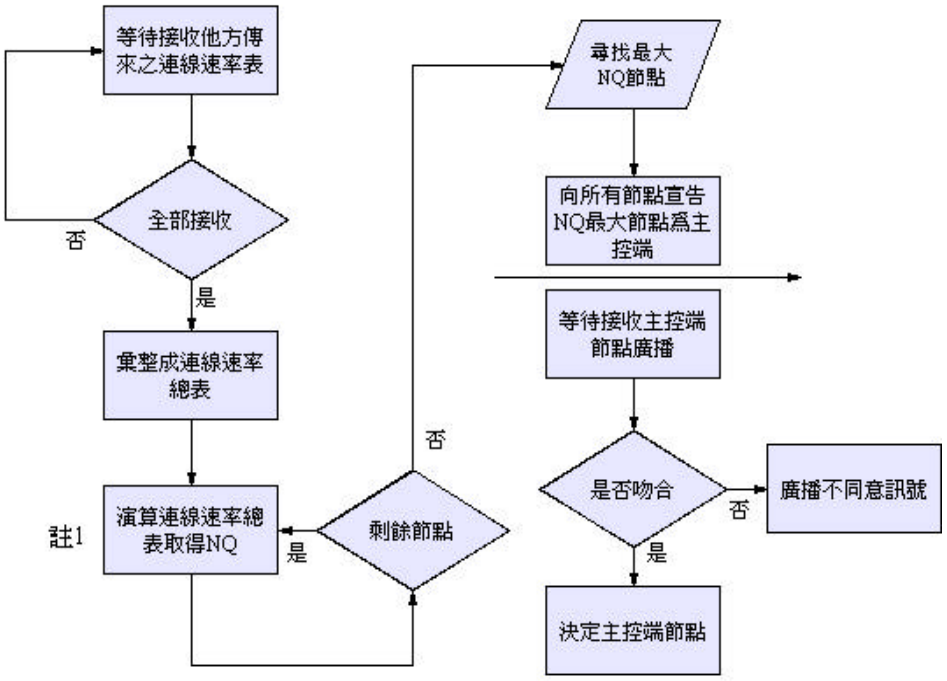


圖 2.4 建立本地端連線速率表演算流程



註1
NQ = ((自己對每個節點的上傳速度總和+所有節點對自己的上傳速度總和) / (總結點數*2)) *(全雙工或半雙工 註1) * 節點平衡因素 註2

註1。全雙工 = 1 , 半雙工 = 0.5
註2。節點平衡因素 = 上傳頻寬 / 下傳頻寬 (if >1 =>1/(上傳頻寬 / 下傳頻寬))

圖 2.5 整合連線速率表並計算節點品質因素(NQ)之演算流程

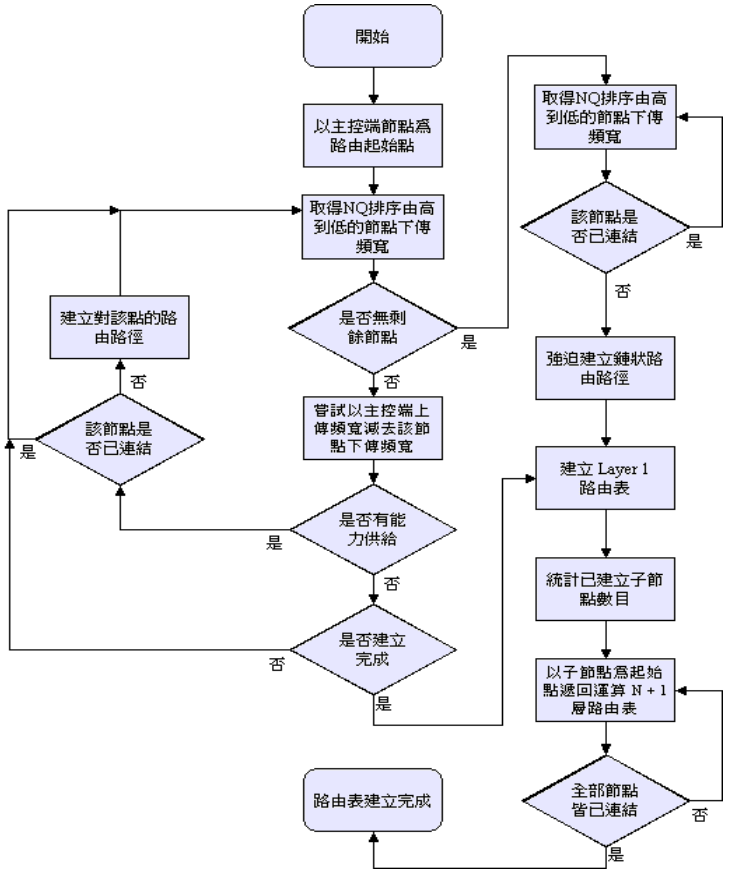


圖 2.6 路由演算法流程圖

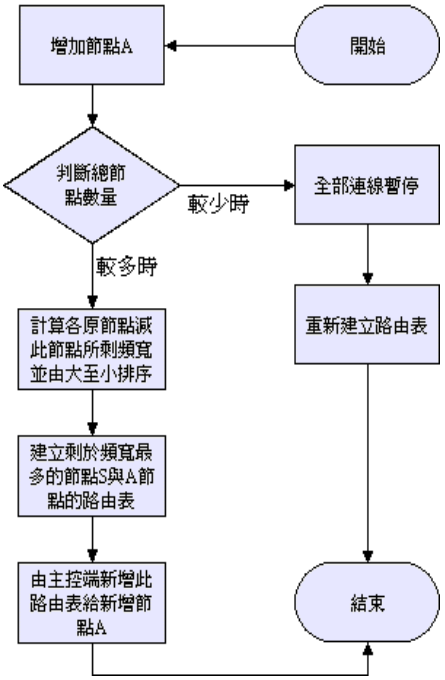


圖 2.7 路由表修正演算法流程 – 增加節點

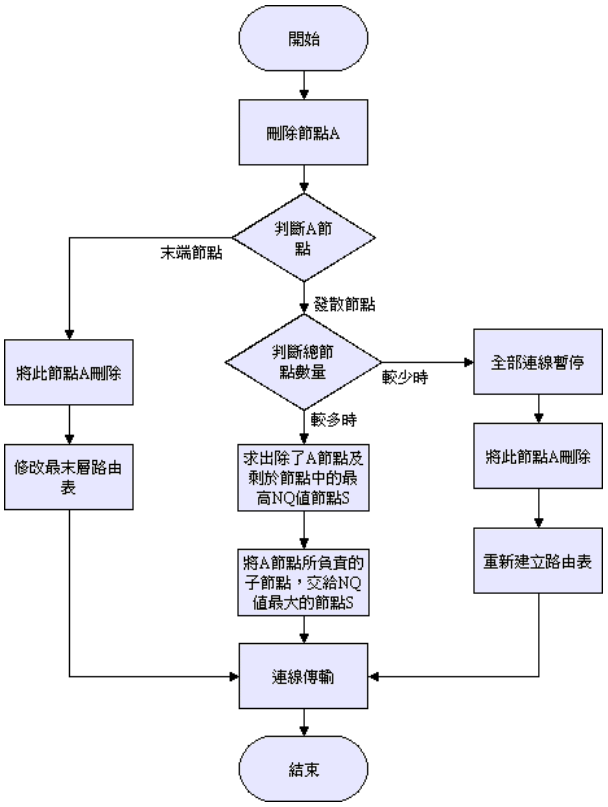


圖 2.8 路由表修正演算法流程 – 刪除節點

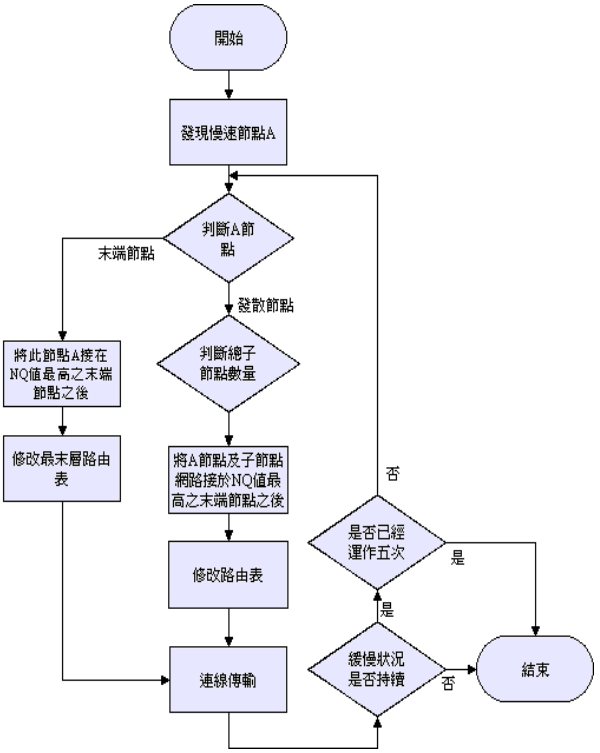


圖 2.9 路由表修正演算法流程 – 慢速節點處理

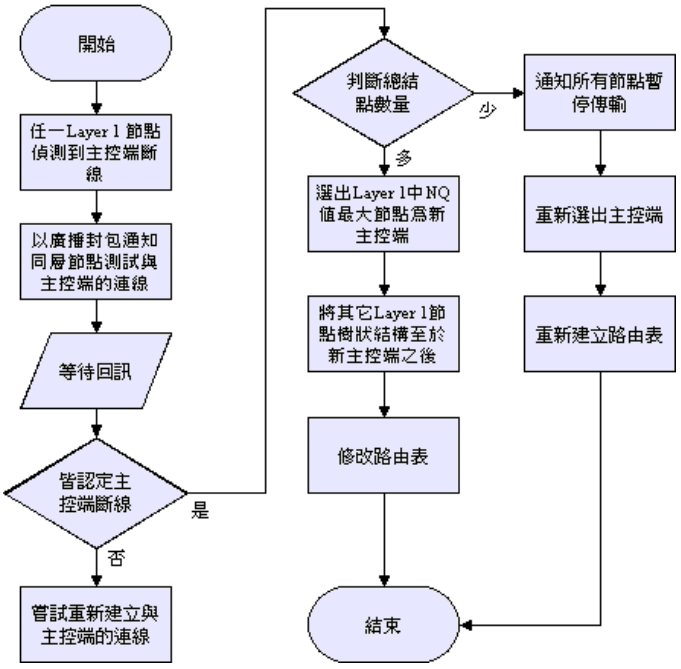


圖 2.10 路由表修正演算法流程 – 主控端斷線處理

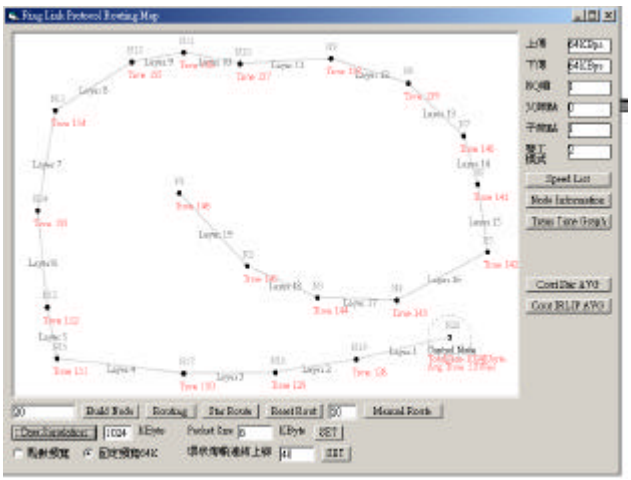


圖 3.1 模擬環境 1之路由網路圖

.

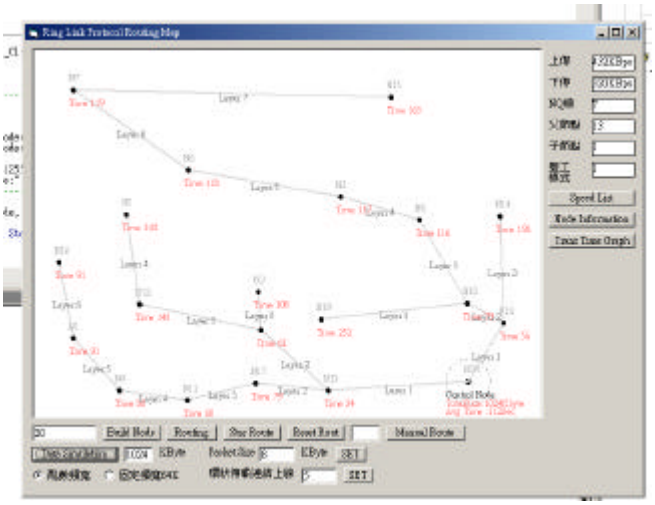


圖 3.2 模擬環境 2 之路由網路圖

表 2.1 傳輸封包格式

欄位名稱	欄位長度	說明
Packet_Start	4Byte	封包起始標記
Transfer_SN	4Byte	傳輸編號
Total_Size	4Byte	傳輸總容量
Packet_SN	4Byte	封包編號 (流水號)
Key_SN	4Byte	加密欄位
Command	4Byte	指令區域
Route_Table	128Byte	路由表
Data_Area	依傳送大小而定	資料區段
Packet_End	4Byte	封包節尾標記

表 3.2 亂數決定節點外在因素之模擬結果

傳統星狀網路	效率型網路群播協定
Test 1 : 812Sec	Test 1 :205Sec
Test 2 :691Sec	Test 2 :307Sec
Test 3 :529Sec	Test 3 :205Sec
Test 4 :635Sec	Test 4 :307Sec
Test 5 :957Sec	Test 5 :205Sec
Test 6 :1056Sec	Test 6 :307Sec
Test 7 :1553Sec	Test 7 :205Sec
Test 8 :649Sec	Test 8 :307Sec
Test 9 :660Sec	Test 9 :205Sec
Test 10 :887Sec	Test 10 :307Sec
Test 11 :653Sec	Test 11 :205Sec
Test 12 :649Sec	Test 12 :307Sec
Test 13 :512Sec	Test 13 :205Sec
Test 14 :1024Sec	Test 14 :307Sec
Test 15 :574Sec	Test 15 :205Sec
Test 16 :709Sec	Test 16 :307Sec
Test 17 :1016Sec	Test 17 :205Sec
Test 18 :535Sec	Test 18 :307Sec
Test 19 :944Sec	Test 19 :205Sec
Test 20 :577Sec	Test 20 :307Sec
Average : 781	Average : 256

表 3.3 測速演算機制之各節點測試結果

IP	RTT	Bandwidth	duplex
192.168.0.1	*為主控端節點，不做測試		
192.168.0.2	46ms	5434 Kbps	Full
192.168.0.3	49ms	5102 Kbps	Full
192.168.0.4	51ms	4901 Kbps	Full
192.168.0.5	36ms	6944 Kbps	Full
192.168.0.6	29ms	8620 Kbps	Full
192.168.0.7	34ms	7352 Kbps	Full
192.168.0.8	42ms	5952 Kbps	Full
192.168.0.9	45ms	5555 Kbps	Half
192.168.0.10	31ms	8064 Kbps	Full

表 3.1 所有節點及頻寬等資訊列表
(此列表由模擬程式以亂數產生)

編號	緩慢	NQ 值	NQ 排名	下傳頻寬 Kbps	上傳頻寬 Kbps	工作模式
1	否	115	14	272	400	半雙工
2	否	191	9	464	272	全雙工
3	否	188	10	192	192	全雙工
4	否	149	12	464	416	半雙工
5	否	126	13	496	320	半雙工
6	否	104	15	128	464	半雙工
7	否	97	17	240	176	半雙工
8	否	265	6	208	336	全雙工
9	否	400	4	416	416	全雙工
10	否	302	5	240	416	全雙工
11	否	166	11	256	304	半雙工
12	否	212	8	352	352	半雙工
13	否	484	2	496	496	全雙工
14	否	93	18	96	224	半雙工
15	否	69	19	304	48	半雙工
16	否	99	16	400	96	半雙工
17	否	212	7	320	432	半雙工
18	否	425	3	304	432	全雙工
19	否	58	20	96	16	全雙工
20	否	536	1	384	384	全雙工