基於網格樹拓樸的 P2P 架構及其效能評估
Performance Evaluation of an MCT-Based Peer-to-Peer Architecture

李政宏　彭思淵　呂紹偉
國立臺灣海洋大學
電機工程學系
E-mail:{kristof, uftea, swleu}@ntou.net

詹景裕
國立臺北大學
電機工程研究所
E-mail:gejan@mail.ntpu.edu.tw

***摘要*—網格樹(Mesh-Connected Tree, MCT)是一種具有階層性結構的網路拓樸，利用此種網路拓樸所建構的點對點架構具有良好的擴充性與時間複雜度，因此非常適合應用於結構化點對點網路。因此在本文中，我們利用實體距離來安排每個新進節點的位置，以縮短鄰近節點取得資料所需的時間，避免兩個鄰近節點可能要繞過很長的搜尋路徑才能相互取得資料的情形。另外，我們採取二元樹搜尋演算法讓需求者搜尋所需檔案，因此搜尋的時間複雜度僅為 $\log n$ 。在效能評估方面，我們針對網格樹、Chord 架構，以及同樣以階層性結構為基礎的 Grapes 等三種 P2P 架構進行詳細的分析比較。我們使用四個評估參數，分別是延遲時間(delay time)、抖動率(jitter)、封包遺失率(packet losses)與傳輸量(throughput)。結果顯示，基於網格樹的 P2P 架構，在相同的環境參數下，其整體的效能為三者最高，因此檔案傳輸的效能應該可以預期是三者之中最為高的。**

***關鍵詞*—網格樹、延遲時間、抖動率、封包遺失率、傳輸量。**

Abstract

The Mesh-Connected Tree (MCT) is a hierarchical network topology formed by joining two layers of network structures. The top layer is a mesh with each node being the root of a binary tree. Previous study has shown that a P2P network based on the MCT topology is easy to expand and able to distribute search messages through out the whole structure efficiently. However, if an MCT-based P2P network assigns location for each new peer node by using a *distributed hash table* (DHT) based algorithm, then search efficiency will be hard to improve, because two physically nearby peers may be allocated positions far apart in the MCT hierarchy. To solve this problem, we use ping response time to estimate network distance such that peers in proximity will not waste time going through a long search path to exchange information with each other just because they have been unwittingly allocated too far apart. Furthermore, we propose a binary search method for a requester to locate the potential provider of files of interest; a $\log n$ search time is thus achieved. We compare the performance of the proposed MCT-based P2P network with two other previously proposed P2P architectures, the Chord and the Grapes. Simulation results show that the proposed system not only has shorter average delay and lower jitter for packet transfer, but also has fewer packet losses and higher throughput. These results also indicate the potential of the MCT topology to be the foundation of a P2P network that is both efficient and reliable.

***Keywords* – Mesh-connected tree, delay time, jitter, packet losses, throughput.**

一、導論

點對點[1][10][11][21]的資源分享方式與傳統集中式的資源分享架構有非常顯著的差異。傳統的資源分享是將資源提供者(Provider)所釋出的資源全部集中於一台伺服器上，需求者(Requester)則利用搜尋功能取得所需要的資源。點對點網路不需要集中式的中央控管機制即可讓系統之間直接進行資訊和服務的分享，這些資源和服務包括檔案共享(File Sharing)、即時訊息

(Instant Messaging)、以及分散式運算(Distributed Computing)等。在點對點網路架構下，任何節點都可以是服務需求者也可以是服務的提供者。因此，傳統主從式架構容易出現的伺服端負載過高以及頻寬需求過大的問題都可以獲得抒解。

由於點對點架構沒有集中式的資源管理，所以在資源的搜尋上較無效率[2][22]，比較容易發生搜尋不到已經存在的資源或搜尋回報的結果不正確的情況。在 P2P 網路架構中，網路上的節點必須先加入 P2P 網路社群，成為 P2P 網路社群的成員(peer)後，才能透過點對點網路服務搜尋或下載所需的資源。在點對點網路架構中，服務可分為搜尋與下載兩種。當一個 peer 需要資料時，會發出搜尋的訊息到 P2P 社群中搜尋所需的資料，然後根據回報訊息直接向擁有該項資料的 peer 要求下載。由於資料的分布遍及所有節點，相較於主從式架構只需向伺服端查詢的運作方式，P2P 網路資料搜尋的複雜度因而大幅提高。目前常用的點對點系統都分別提出不同的搜尋機制，讓 peer 可以快速的找到所需要的資料。依據所採用的搜尋機制，點對點網路架構可區分為下列三種不同的運作方式：[4][17][20][19]

- Centralized Directory Model
- Unstructured Model
- Structured Model

以 Centralized Directory Model 運作的點對點網路架構，如 Napster，BitTorrent 等，必須建立一個經常更新的中央目錄管理伺服器(Centralized Directory Server)，記錄 peer 的 IP 位址及可分享的檔案名稱，讓所有 peer 搜尋所需的檔案。這種看似 Client/Server 架構的點對點網路架構，由於伺服端本身只提供所有 peer 可供分享的資料及位址的資訊，並不直接讓 peer 下載，因此可以有效的降低伺服端頻寬需求。不過，由於中央目錄管理伺服器的容量還是有限，因此限制了社群的擴展性。此外，當中央目錄管理伺服器發生問題時，更會造成整個點對點網路社群無法運作，使得此種架構的容錯性(Fault Tolerance)相對較低。

Unstructured Model 與 Structured Model 皆屬於分散式的作業方式。採用 Unstructured Model 的點對點網路架構，如 Gnutella 與 Freenet，皆不需要中央目錄管理及資料放置的機制，也不需要控管拓樸架構，加入社群的方式是連接社群中任意鄰近的節點。搜尋資料的方法是利用 Flooding 的方式傳送訊息，透過鄰近連接的節點相互發送訊息(Store and Forward)的方式來搜尋所需的資料。雖然這種成員可以隨時動態加入或離開的搜尋方式有很大的彈性，可以有效舒緩搜尋瓶頸的問題，但其缺點是會產生大量的搜尋訊息造成網路擁塞，並且仍有搜尋不到資源的可能性。

分散式雜湊表(Distributed Hash Table, DHT)和各種疊加網路(Overlay Network)拓撲成為目前點對點網路系統的基礎。在分散式雜湊表的一致性雜湊(Consistent Hashing)下，所有節點均能配置在疊加網路中而達到負載平衡。此外，為了提高搜尋效率，結構化點對點網路系統會依照不同的疊加網路，而設計初步不同的繞送(Routing)和配置(Allocating)演算法。因此有許多論文提出 Structured Model 的搜尋演算法，如 CAN [14]、Pastry [15]與 Chord [3][18]等。此外由於分散式雜湊表大量運用於結構化點對點網路架構(Structured Peer-to-Peer Network)，利用此一演算法將資源與服務平均分散於整個網路系統，讓 peer 依照有規則的搜尋方式來搜尋資料，避免所發送的訊息造成網路訊息的氾濫，並增加社群的擴展性。結構性點對點網路架構中的分散式雜湊表，提供傳繞與位置配置的資訊。每一個網路上的節點都是對等的，並儲存一部份其他節點的資訊，幫助傳繞的達成。此外，在高動態的網路下，節點頻繁的加入與離開系統會對 DHT 的維護帶來極大的負擔。

Chord 是一種非常著名的點對點架構，利用 finger table 來達成搜尋的方法，其時間複雜度只有 $\log n$，且具有良好的擴展性，因此相當受到重視。此外，在 Chord 網路中也要避免因為節點的加入或移除，所造成的錯誤。相較於 Chord，由於 Grapes[16]是一種階層式的點對點架構，且具有 $\log\sqrt{n}$ 的 Logical Lookup Hops，因此搜尋的時間將會短於 Chord，但此一情況必須建立在 Grapes 的上下兩層都採用 Chord 演算法，方能有此時間複雜度。因為當需求者搜尋某一檔案資料時，假設所要搜尋的檔案在 Chord 的最遠處並且 Chord 的規模也很大時，在真實的網路架構下，其所花費的時間將會很長。但由於 Grapes 的每個節點都是利用實體距離所配置，兩節點之間的距離都很相近，因此 Grapes 將有助於縮短搜尋時間。由於在結構化的 P2P 中，其資料是藉由 DHT 所配置，因此需求者必定可以找到，差別只在於搜尋時間的長短與封包發送數量是否會造成網路壅塞。因此如何在最短的時間內利用最少的封包搜尋到最多的資料，將是未來 P2P 的重要議題。

本論文利用網路鏈結的實體距離來估計需求者與提供者之間的 RTT (Round-Trip Time)。實體距離的估算可藉由 Ping 或其他量測方法 取得與其他節點之間的傳送時間，利用這一時間來作為一個節點被允入 MCT 後位址配置的主要依據。我們將針對整體的封包延遲時間、抖動情形、遺失率與傳輸量對 MCT 架構進行模擬分析，並提出與搜尋演算法，此外也與另外二個同屬 Structured Model 的 P2P 架構，即 Chord 和 Grapes，做詳細比較。本文第二節將介紹 MCT 的設計，第三節是參數的定義，第四章是模擬，第五章是效能分析，最後是結論與未來工作。

二、網格樹(MCT)

原始的 MCT [6]由於有良好的節點利用率、高擴展性、以及較少的跨線，因此可用於 VLSI 的佈局設計。Li 等則將該架構應用於混合式 P2P 的傳輸上[9]，利用該架構的階層特性[5][7][8]，分析其延遲時間，抖動率，傳輸量與封包遺失率，並得到不錯的效能表現。

MCT 是一個具有自我組織的階層式網路架構，由二種網路拓樸組成，上層由一個網格架構(mesh)構成超網路[12]，下層則是由二元樹形成子網路，而每一子網路的樹根即為形成超網路的各個節點。超網路中的節點負責管理與配置其下的子節點，而下層二元樹的子節點則是經由配置演算法所允入的新進節點[5]，如圖一所示。我們提出利用實體距離來作為每個節點進入到 MCT 的依據，因此每個節點彼此之間實際距離都很接近，並利用二元搜尋樹的演算法安排每一個需求者的位址。由於建構 MCT 是採用二元搜尋樹演算法，因此，該架構的時間複雜度為 $\log n$，倘若整體架構負載平衡時則其複雜度為 $\log\sqrt{n}$，相較於同樣是階層式架構的 Grapes 而言，整體搜尋效能提高很多。緊接著我們更進一步討論 MCT 的架構。

在演算法的初始階段，使用者必須先定義進入 MCT 的臨界時間(Threshold Time)，以便每個節點在進入 MCT 時有一個比較的依據。以下分別針對於節點的進入、離開與搜尋演算法詳加說

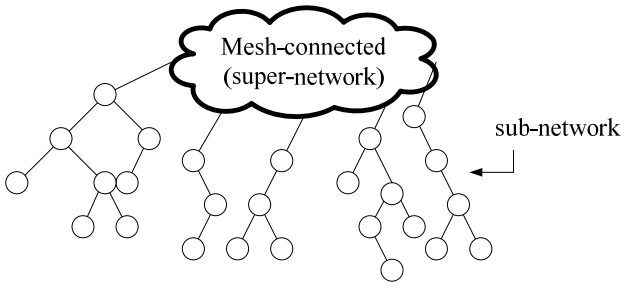

圖 1 一個由二元樹所建構成的階層式 MCT

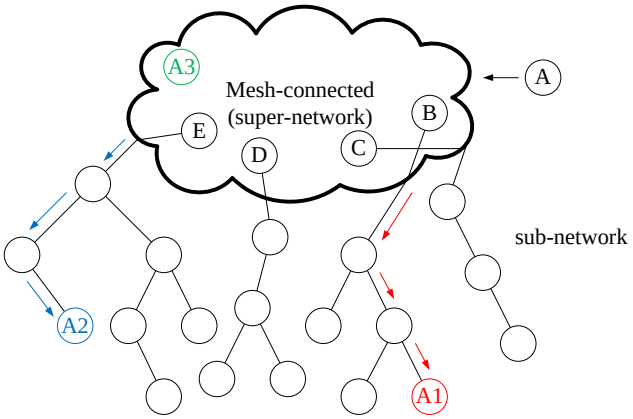

圖 2 在 MCT 架構中節點的插入

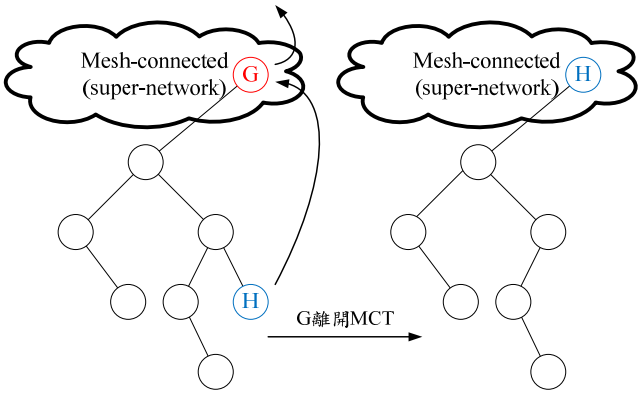

圖 3 超節點 G 離開 MCT 架構的範例

明。

a. Node Insertion

一個新的節點若要進入 MCT，必須接觸至少一個稱為 bootstrap 的 MCT 節點。新節點首先檢查與 bootstrap 的實體距離，假如實體距離較臨界值為短，新節點將會被排在 bootstrap 的子網路內，並利用二元搜尋樹演算法指定位置予新節點。反之，若實體距離大於或等於所給定的臨界值，則新節點會被安排在超網路。若新節點在超網路上沒有找到合適的位置，則成為一個領導節點，提供往後其它新進節點的比較之用。如圖二所示，當新節點 A 要加入 MCT 時，須藉由 bootstrap 節點 B 的幫助方能進入。假設節點 A 與節點 B 的實體距離小於臨界值，則 A 會被安排至 B 的子網路內，如圖中的節點 A1。反之，若節點 A 與節點 B 的實體距離大於臨界值時，則繼續與超網路中的其他節點進行比較。在圖中，因為節點 A 與節點 C 或 D 之間的實體距離均大於臨界值，因此節點 A 不會被配置到節點 C 或節點 D 的子網路。假設節點 A 與節點 E 的實體距離小於臨界值，則 A 會被安排在節點 E 的子網路內，如圖中的 A2。假設沒有合適的位置給予節點 A，則節點 A 將成為一個超節點，如圖中的 A3，以供後續進入的節點比較之用。演算法描述如下：

```
節點加入演算法()
initialization
  選定臨界值
begin
  if （新節點與起始節點的實體距離 ≥ 臨界時間）
    將新節點配置在超網路
    if（新節點與超節點的實體距離 ≥ 臨界時間）
      形成領導節點
  else
    將新節點配置在子網路
end
```

b. Node Deletion

當節點要從 MCT 離開時，由於節點所在位置的不同，將會有不同的作法。我們分成兩個部分討論：第一部份：倘若需求者是超節點時欲離開 MCT，因為我們在子網路所用的演算法為二元搜尋樹，因此離開的空缺將會由該子網路中最右邊的節點來擔任超節點，如圖三所示。第二部分：倘若需求者在子網路中欲離開 MCT 時，來下的空缺則由需求者的右節點來取代。如圖四所示。

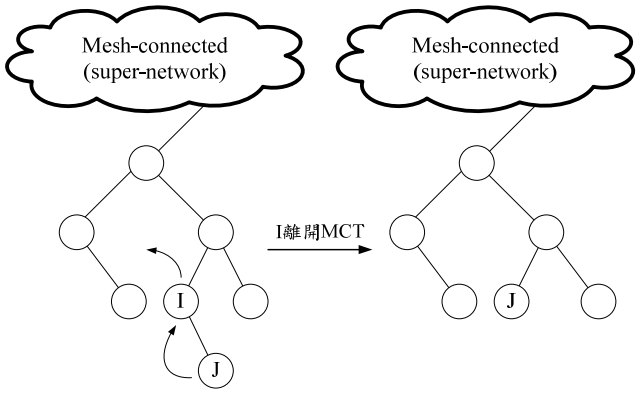

圖 4 子節點 I 離開 MCT 架構的範例

第三部分：若超網路下只有一個子節點時，當超節點離開時，則由該子節點擔任超節點，演算法如下。

節點離開演算法()

```
begin
  if 節點在超網路
    if 此節點只有一個子節點
      由子節點擔任超節點
    else
      由子網路內最右邊的子節點擔任超節點
  else //節點在子網路
    離開的位置由節點的右節點取代
end
```

c. Search Algorithm

最後，我們提出 MCT 的搜尋演算法。在 MCT 架構中的每個節點都與 bootstrap 比較過後，被一一安排在 MCT 架構內。此時，架構內的每個節點都可以成為服務的需求者也可成為服務的提供者。需求者所在的位置將會影響搜尋的策略。由於在結構化的點對點網路中的資料必定能找到，因此資料所在的位置，將會是搜尋的一個依據。當需求者提出請求時，由於知道提供者的位置，因此在搜尋時會與提供者比較 ping 值的大小，若比提供者大，則提供者的位置會在需求者的前面，反之則在後面。判斷完成之後，緊接著利用網格的搜尋，找到提供者的領導節點後，在利用二元搜尋法找到提供者。整個搜尋的演算法如下所示。

搜尋演算法()

```
begin
  if （提供者與需求者同在一個子網路下）
    使用二元搜尋法找到提供者
  else
    搜尋提供者所屬的超節點位置
    if （超節點不是提供者）
      使用二元搜尋法在該超節點下找到提供者
end
```

最後，我們比較與其他兩個同樣是結構化的 P2P 架構，雖然都具有相同的時間複雜度，但實際去評估這三種架構在真實網路的傳輸情況，就會發現這三種架構在傳輸時的差異性，我們在下一節中討論。

三、參數定義

本節根據平均延遲時間、抖動率、封包遺失率與傳輸量等四個參數來評估前述三種架構的網路傳輸效能[13]。各參數的定義詳述如下：

- 平均延遲時間(Average delay)

我們定義平均延遲時間是指封包在到達時間與傳送時間之間的差值，我們用一數學式來表示整個網路的平均延遲時間 D 如下：

$$D = \frac{\sum_{i=1}^{N} d_i}{N}$$

其中 d_i 是指全部延遲時間的總和，N 表示所有節點間測量值的數量。

- 抖動率(Jitter)

由於網路的流量隨時都在變化，因此當流量變大時，許多從節點發出的封包就必須在佇列中等待傳送，因此造成每個封包從傳送節點到目的節點的時間就會不一樣，這個不同的差異就是抖動率，抖動率越大，代表網路越不穩定。我們可以用一數學式來表示整個系統的抖動情形，如下：

$$Jitter = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(d_i - D\right)^2}$$

- 封包遺失(Packet losses)

由傳送節點到目的節點之間,因為封包碰撞，佇列溢位或 TTL 等問題而造成封包在傳輸途中遺失。我們用一數學式來表示整體封包遺失 L，

$$L = \sum_{t=0}^{t=n}\left|\left(P_r - P_s\right)\right|$$

其中，P_r 代表封包接收數量，P_s 代表封包傳送數量，n 表示時間。

- 傳輸量(Throughput)

Throughput 是指在單位時間內節點間封包的傳輸量。假設傳輸的通道(Channel)壅塞，則傳輸量將會下降。一般來說，在相同的參數下，產出量越高的網路，其整體的效能越好。

在這一段中，我們提出我們模擬所需要的參數設定，為了能夠精確的模擬出三種網路架構的平均延遲時間、抖動率、封包遺失率與傳輸量，我們設定以下的參數，如表一所示，來評估 MCT、Grapes 與 Chord。此外，我們所使用的模擬軟體為 NS2 (Network Simulator version 2)。

四、模擬結果與效能分析

我們分析這三種架構下在真實網路的模擬情形，由於網路傳輸時不可能只有單一傳送端與接收端，因此為了能夠精確模擬出整體的效能，三個網路架構各有 19 個傳輸路徑，並於同一時間開始傳輸，讓整個網路處於忙碌狀態，也比較符合實際的網路狀態，每個路徑的模擬時間為 100 ms，結束時間是 105 ms。

首先，我們分析延遲時間，經由模擬後，得到如圖五、圖六與圖七所示。我們發現整個 Grapes 與 Chord 的延遲時間相較 MCT 來的非常嚴重，在模擬時間一開始的時候，該架構即有延遲時間現象的產生，封包在架構內開始被傳送，因整體網路處於忙碌狀態，因此延遲時間非常的

表 1 參數定義

名詞	定義
節點數	為了能夠精確模擬出趨近實際網路的效能，我們所設定的節點數為 100 個節點。如此也避免因為節點數量過大，而造成分析不易。
臨界值	新節點要加入 MCT 時的臨界值在此設定 100 ms。
頻寬	節點之間的連線頻寬為 2 MB。
佇列	連線之間的佇列大小為 10。
連線	節點之間的連線其延遲時間為 10 ms。
傳輸模式	我們建構在 TCP/IP 的網路下，採用 FTP 的傳輸模式。
封包大小	節點傳送的封包大小為 1000。
TTL (Time-to-Live)	此處設定為 32。

嚴重，我們更進一步分析模擬後的數據，如表一所示，MCT 的平均延遲時間較 Grapes 少 1.31 倍的時間，較 Chord 少 1.68 倍的時間，因此整體的延遲擺動範圍也較 Grapes 與 Chord 來的小。因此在網路傳輸上，傳輸品質將會較佳。由於模擬的時間只有 100 ms，所以整體的延遲時間變動誤差較小，但若是將模擬時間拉長，則變動的誤差將會更加明顯，並且呈線性成長。

抖動率的比較如圖八、圖九與圖十所示。MCT 因為延遲時間較短，所以所造成的抖動範圍相對較 Grapes 與 Chord 小。模擬得到的數據如表二所示，二者的擺動幅度大小相似。依數據，MCT 擺動幅度較 Grapes 與 Chord 分別小 1.40 與 1.62 倍，可見在傳輸品質的穩定程度上，MCT 優於 Grapes 與 Chord。此外，我們分析其封包在傳送時的遺失率，在這三個架構內皆有 19 條傳送路徑，傳送後的結果如表三所示，在 MCT 架構中，一共送出了 2,124,007 個封包，遺失了 572 個封包，遺失率為 0.000269，在 Grapes 的架構中一共送出 1,430,981 個封包，遺失了 3,298 個封包，遺失率為 0.002304，在 Chord 的架構中一共送出 3,585,061 個封包，遺失了 11,201 個封包，遺失率為 0.003124。由模擬結果可以得到，MCT 的封包遺失率明顯低於 Grapes 與 Chord。因此，若將此架構實現於實際的網路系統，MCT 的傳輸品質預期將會優於 Grapes 與 Chord。

最後，在 Chord 的架構下，19 個傳輸路徑一共經過 259 個 link，其傳輸量為 101,057.78 kbps。在 Grapes 的架構下，經過 118 個 link，其傳輸量為 61,411.57 kbps，最後在 MCT 的架構下，經過 102 個 link，其傳輸量為 99,488.59 kbps。整個傳輸量的結果如表四所示。在 Chord 架構內雖然有較多的 link，其傳輸量應該會較多，但平均的傳輸量卻只有 390.18 kbps，在 Grapes 架構內，平均的傳輸量為 520.43 kbps。最後在 MCT 架構內，平均的傳輸量為 957.38 kbps。最後由模擬的

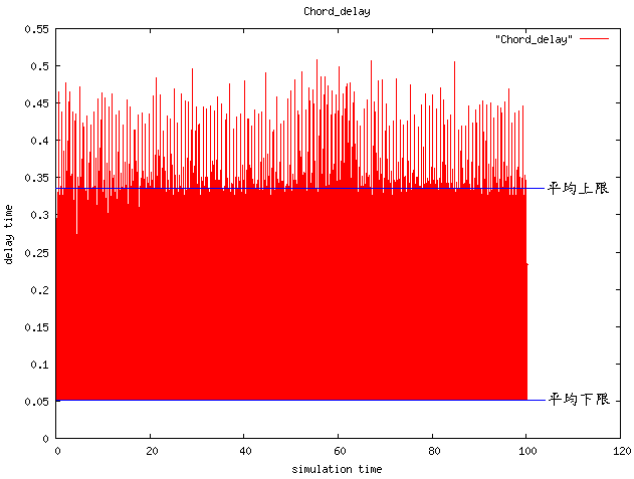

圖 5 Chord 的延遲時間變化情形

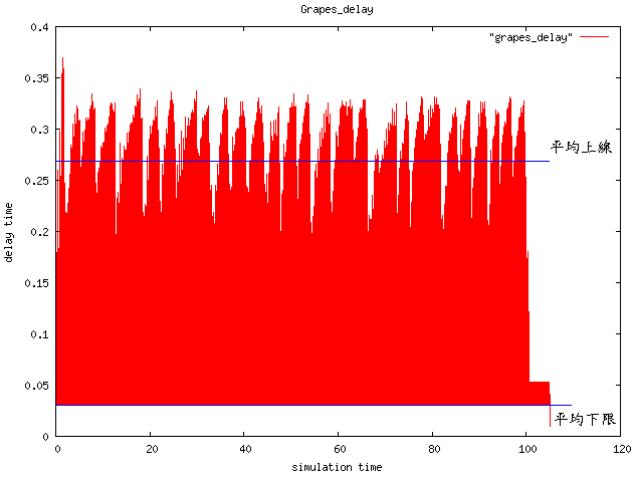

圖 6 Grapes 的延遲時間變化情形

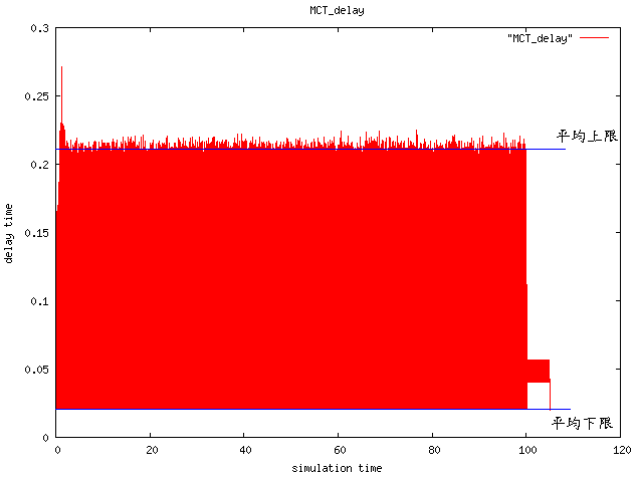

圖 7 MCT 的延遲時間變化情形

表 2 三種架構的延遲時間比較

	平均上限	平均下限	變動範圍
Chord	0.337936	0.050872	0.388808
Grapes	0.274630	0.029281	0.303911
MCT	**0.210254**	**0.020533**	**0.230788**

表 3 三種架構的抖動變化比較

	平均上限	平均下限	變動範圍
Chord	0.013002	–0.003277	0.016279
Grapes	0.011271	–0.002762	0.014033
MCT	**0.008837**	**–0.001221**	**0.010058**

表 4 三種架構的封包遺失率

	封包遺失率		
	總封包	遺失封包數	百分比
Chord	3585061	11201	0.003124
Grapes	1430981	3298	0.002304
MCT	**2124007**	**572**	**0.000269**

表 5 三種架構的傳輸量比較

	路徑	傳輸量	平均傳輸量
Chord	259	1010572.78	390.18
Grapes	118	61411.57	520.43
MCT	**102**	**99488.59**	**957.38**

結果可以得到在相同的環境參數下，MCT 的傳輸量分別較 Chord 與 Grapes 高 2.45 倍與 1.84 倍。因此可以得知 MCT 在真實網路的效能將會優於 Chord 與 Grapes。

我們綜合之前的文獻，針對 MCT、Chord 與 Grapes 依據上述四個參數，延遲時間、抖動變化、封包遺失率與平均吞吐量來評估一個網路的整體效能。由表二至五中可以得知，在相同參數的環境下，利用網路模擬軟體來模擬真實的網路環境，雖然這三種架構的時間複雜度都一樣，但詳細分析其四個參數，就會發現其中 Chord 的架

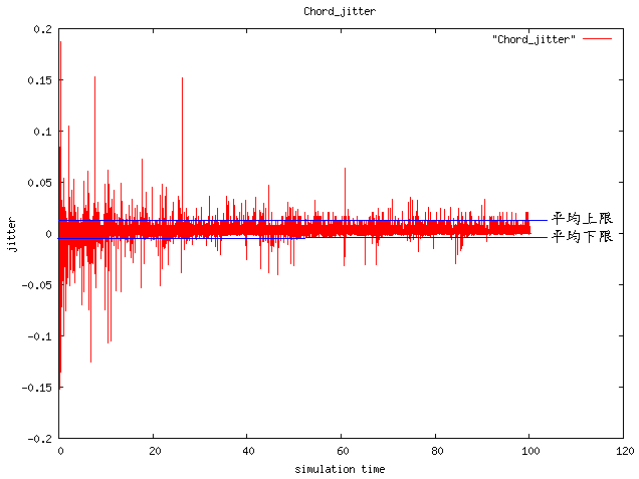

圖 8 Chord 的抖動變化情形

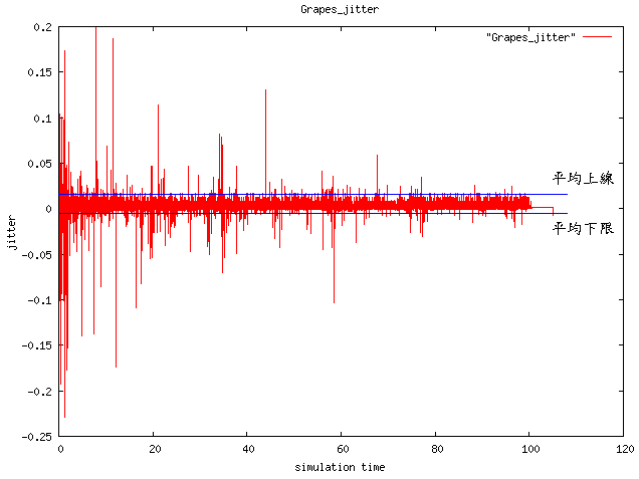

圖 9 Grapes 的抖動變化情形

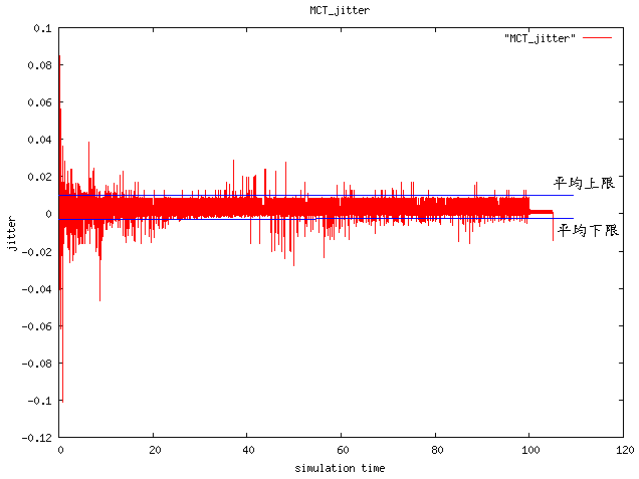

圖 10 MCT 的抖動變化情形

構，其網路架構較接近 Ring，在傳輸時由於只有單一個路徑，其他的資料在傳輸時也有可能要使用到這一路徑，因此在傳輸時網路很容易壅塞，進而造成封包遺失，此外也會造成因延遲而產生抖動的變化，影響傳輸的品質。而 Grapes 的四個參數的表現皆優於 Chord，其每個節點的分支度較多，資料在傳送時有較多的路徑選擇，在多組傳輸時，可以避免在高速傳輸時網路的壅塞與封包的遺失。最後，MCT 架構在各方面皆有不錯的表現，在多組傳輸時，整體的效能是這三種架構中最好的，因此如果將 MCT 建構至真實網路時，其傳輸上的品質將非常的穩定。

五、結論與未來工作

本文利用實體距離來建構一個結構化 P2P 網路，並利用二元搜尋樹的特性降低搜尋的時間複雜度。除了比較同屬層級式架構的 Grapes 與我們所提出的網格樹架構之外，我們也進一步分析 Chord 架構在真實網路中的效能表現。由模擬結果與分析比較可知，與效能表現息息相關的各項重要變量，如平均延遲時間、抖動變化量、封包遺失率與傳輸量等，MCT 的表現均優於 Grapes 與 Chord。未來可能的發展方向之一是將 MCT 架構下的成員分群，進一步提昇資源搜尋和傳輸的效率。

參考文獻

[1] J. F. Buford, H. Yu, and E. K. Lua, *P2P Networking and Application*, Morgan Kauffmann, 2009.

[2] A. Crespo and H. Garcia-Molina, "Routing indices for peer-to-peer systems," *Proceedings of the* 22*nd International Conference on Distributed Computing Systems*, 2002, pp. 23-32.

[3] F. Dabek, E. Brunskill, M. F. Kaashoek, D. Karger, R. Morris, I. Stoica, and H. Balakrishnan, "Building peer-to-peer systems with Chord, a distributed lookup service," *Proceedings of the* 8*th Workshop on Hot Topic in Operating Systems*, pp. 81-86, 2001.

[4] M. B. Dumas and M. Schwartz, *Principles of computer networks and communications*, Prentice Hall, 2009.

[5] H. C. Hsiao and C. T. King, "A tree model for structured peer-to-peer protocols," *Proceedings of the* 3*rd IEEE/ACM International Symposium on Cluster Computing and the Grid*, pp.336- 343, 2003.

[6] G. E. Jan, S.-W. Leu, C. H. Li, and W. R. Liou, "Yet another mesh-connected tree parallel architecture," 2004 *International Computer Symposium*, pp. 1358-1361, Taipei, Taiwan, 2004.

[7] H. T. Kung and C. H. Wu, "Hierarchical Peer-to-Peer Networks," Technical Report IIS-TR-02-015, Institute of Information Science, Academia Sinica, Taiwan, April 2001.

[8] R. Kurmanowytsch, M. Jazayeri, and E. Kirda, "Towards a hierarchical, semantic peer-to-peer topology," *Proceedings of the* 2nd *International Conference on Peer-to-Peer Computing*, pp. 167-168, 2002.

[9] C. H. Li, S.-W. Leu, and G. E. Jan, "A Mesh-Connected Tree Based Hybrid Peer-to-Peer Network", 2007 *IEEE TENCON*, Taipei, Taiwan, 2007.

[10] G. Li, "Peer-to-peer networks in action," *IEEE Internet Computing*, Vol. 6, Issue 1, pp.37-39, 2002.

[11] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu, Peer-to-Peer Computing, Technical Report, HPL-2002-57, HP Lab, 2002.

[12] S. Murthy and A. Sen, "A Peer-to-Peer Network Based on Multi-Mesh Architecture," *Proceedings of IEEE GLOBECOM*, pp.3840-3844, 2003.

[13] N. Olifer and V. Olifer, *Computer Networks*

principles, technologies and protocols for network design, John Wiley & Son, 2006.

[14] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Schenker, "A Scalable Content-Addressable Network," *Proceedings of the* 2001 *conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 161-172, 2001.

[15] A. I. T. Rowstron and HP. DruschelH, A Pastry: Scalable, "Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, pp. 329-350, 2001.

[16] K. Shin, S. Lee, G. Lim, H. Yoo, and J. S. Ma, "Grapes: Topology-based hierarchical virtual network for peer-to-peer lookup services," *Proceedings of the International Conference on Parallel Processing Workshops*, pp.164-195, 2002.

[17] R. Steinmetz and K. Wehrle (Eds.), *Peer-to-peer systems and applications*, LNCS 3485, Springer, 2005.

[18] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Transactions on Networking*, Vol. 11, Issue 1, pp. 17-32, 2003.

[19] R. Subramanian and B. D. Goodman, Peer-to-peer computing: *The Evolution of a Disruptive Technology*, IDEA group publishing, 2005.

[20] S. A. Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies," *ACM Computing Surveys*, Vol. 36, Issue 4, pp. 335-371, 2004.

[21] J. Wu, *Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks*, Auerbach Publications, 2005.

[22] B. Yang and H. Garcia-Molina, "Improving search in peer-to-peer networks," *Proceedings of the* 22*nd International Conference on Distributed Computing Systems*, pp. 5-14. 2002.

[23] BitTorrent, http://www.bittorrent.com/

[24] Freenet, http://freenet.sourceforge.com

[25] Gnutella, http://gnutella.wego.com

[26] NS2, http://www.isi.edu/nsnam/ns/

[27] Napster, http://napster.com