A Structured Overlay for Multi-Attribute Range Queries

You-Fu Yu, Po-Jung Huang, Quan-Jie Chen, Tian-Liang Hunang, Kuan-Chou Lai
Department of Computer and Information Science,
National Taichung University
kclai@mail.ntcu.edu.tw

***摘要*—P2P 網路相關研究中，資源搜尋是一項重要的課題，其中 Chord 為常見的搜尋資源結構。Chord 利用 finger table 記錄節點與後繼節點之間的連結，建立結構化 overlay，使得資源搜尋可在 O(log(*N*))內完成(*N* 為節點總數)。然而，當節點數越多時，Chord 維持 overlay 的花費也越大，而且 Chord 僅針對關鍵字搜尋，限制了 Chord 適用的環境。**

本研究的目標為建構出一個適用於多種應用環境的 overlay。以 Chord 為基礎，將節點屬性與節點 ID 結合，再根據屬性建置出一個多屬性、多環的結構化 overlay。透過多屬性階層式搜尋，不但可縮小搜尋範圍，還可提供多重屬性搜尋的功能。而其多環架構亦可改善系統延展性、減少 overlay 維護的花費。在保留原來 Chord 的特性下，使用雙向連結結構，降低平均搜尋節點數，有效減少搜尋時間，形成一個提供高靈活度，多屬性、範圍式搜尋的 overlay。本研究經 OverSim 模擬實驗顯示確可縮短平均搜尋節點數，減少資源搜尋時間。

***關鍵詞*—**Overlay, Chord, OverSim, Range query, Multi Attribute

一、導論

P2P 網路已發展多年，如何有效搜尋資源一直是許多相關研究的重點。目前的搜尋方式主要分為非結構化與結構化兩種。非結構化的搜尋方法主要以擴播方式傳送訊息，相關研究主要在於改良擴播方式，如何利用最少的訊息量，達到最大的搜尋範圍，但非結構化方法並不能保證一定找得到所需的資源。結構化的方法利用 Distributed Hash Table(DHT)建構出結構化的 overlay，再利用不同的演算法在這個結構化的 overlay 上搜尋所指定的資源，在大部份的 DHT 中，資源 ID 是將資源的名稱 hash 後所得到的，雖然能夠快速、有效的搜尋資源，但是系統必須針對結構化 overlay 延展性與單一關鍵字搜尋的問題加以克服外，還需處理多屬性及範圍搜尋的問題。

許多資源搜尋的研究，只專注於改善 overlay 某一方面的缺點，卻未能提出一個完整的資源搜尋架構。雖然能改善搜尋的效能，但其適用的環境仍然有限。本研究的目的在於構建一個完整的 overlay，不但能提供靈活的資源搜尋模式、又能保留原結構化的特點並改善其缺點，讓此結構化 overlay 能適用於更多不同類型的應用。

本研究以 Chord 為基礎，將 Chord 單一環狀結構，轉變成多屬性、多環結構，不但保留 Chord 結構化的特性，可在有限的搜尋節點數內找到所需資源，且因多環的結構，節點只需記錄所在屬性環的連結，使得維持 finger table 的 overhead 降低，減少整個系統的總連結數，相對的提高了系統的延展性。屬性化的結構則提供了多屬性搜尋的功能，使用者可以搜尋不同屬性的組合，提升了搜尋時的靈活度，再加上使用雙向的連結，可以有效減少搜尋的時間，降低平均搜尋節點數，配合屬性化的結構，還可以達成 multi-attribute range query 的搜尋。

本研究探討了 overlay 對資源搜尋的影響，而搜尋的資源以計算資源為主，每個計算資源都會有 CPU、Memory、HD、頻寬等等固定屬性，根據這些屬性所建立出的 overlay，除了讓原本關鍵字的搜尋更快速外，還可以提供同時指定多種屬性的搜尋，和屬性值介於指定範圍之間的範圍性搜尋。

以下章節簡述 P2P 網路中常見的搜尋方式所面臨的問題並介紹其解決的方法。接著描述相關的背景與研究，及模擬環境 OverSim 的說明介紹。在第三章節則詳細介紹建構本研究所提出的 overlay，並分析該 overlay 對資源搜尋的影響。第四章節介紹相關實驗設計及實驗結果分析。最後介紹結論與未來研究方向。

二、研究背景與相關研究

資源搜尋機制應具備快速、有效率、靈活性與延展性等特性，許多研究 P2P 網路的搜尋策略，亦朝這些方向努力。其中非結構化的搜尋方式如文獻[6] 、[13] 等採用非結構化方式，期望用最少的訊息量，找到最多的資源。在結構化的搜尋方式中，常見的如 Chord、CAN、Tapestry[11] 等，建立結構化 overlay，讓搜尋更有效率。一般來說，結構化的搜尋策略在靈活性與延展性上較佳，而非結構化的搜尋法則能提供快速的搜尋。

在結構化與非結構化這兩種策略中，建立群組為資源搜尋時常用的一種方法；可以將搜尋範圍縮小在某幾個群組中，避免不必要的訊息交換，更快速的找到所需的資源。非結構化的方法中，如文獻[1] 與[2] 分別根據服務群組及使用者的喜好建立群組，利用群組來縮小搜尋時訊息廣播的範圍，但還是無法達到有效的搜尋，或存在可滿足需求的資源卻搜尋不到。而結構化的方法中，如文獻[4] 先將分享相同屬性資源的節點連結起來，形成 local ring，再根據地域性而建立出 global ring，形成一個雙層的架構，但系統必需傳送更多的訊息來找到滿足需求的路徑。文獻[5] 則根據資源的內容建立出多個 context ring，再用一個 super ring 把 context ring 連結起來，這也是一個雙層的架構，但每個節點需要維持更多的連結數，使系統的延展性受到限制。

結構化的搜尋可以提供快速、有效的關鍵字搜尋，但要做到 semantic search 則需要依賴其它的方法，如文獻[3] 中將節點的多種屬性 hash 成節點的 ID，再根據節點 ID 把節點放到相對應的位置，形成一個多環的結構化 overlay，借此達到 semantic search 的功能，但只能達到多屬性搜尋，還無法提供範圍性搜尋。而文獻[9] 則把屬性值轉變成矩陣，利用矩陣運算找到最相近的值，以達到 semantic search 的功能，但也耗費了更多時間在矩陣運算，犧牲了原本結構化 overlay 快速搜尋的特性。

在文獻[10] 中，提出一個階層式結構化的 P2P 系統，可以提供範圍搜尋且有不錯的負載平衡。但因為採用階層式的架構，搜尋訊息在不同階層之間的傳送，反而需要更多的搜尋節點數，才能找到所需的資源，造成搜尋效能的降低。在文獻[12] 中，直接將節點屬性值反應在節點 ID 上，根據不同屬性形成多個屬性環，再將各個屬性環上的 super-peer 連結起來，形成 inter overlay。該文獻並提出利用雙向搜尋來達到範圍搜尋的功能；但在屬性搜尋時，必需搜尋多個屬性環，需要較多的搜尋訊息。在各個屬性環搜尋所得到的節點，並不一定是所求的節點，還得進一步過濾、計算後，才能得到所要搜尋的節點，因而增加額外的運算時間。

OMNet++[7] 是一套以物件為基礎的模擬器，支援 GUI 介面的模擬環境，提供多種模組與基礎架構，還有多種通訊協定，如 IP, UDP, TCP, OSPF, 802.11, IPv6Ethernet, PPP 等等，也提供許多模擬 P2P 網路的協定與模組。而 OverSim[8] 是一套 overlay 基礎架構，提供 OMNet++模擬 overlay 網路，OverSim 也共享 OMNet++的特色，如 GUI 介面，通訊協定，與一些模擬模組等，且可模擬到 10 萬個節點。OverSim 有三種 simple, single host 與 INET 三種 underlay 可以交換，增加了 OverSim 的靈活性與可用性。

三、系統設計

本研究以 Chord 為基礎，在節點 ID 中結合屬性的觀念，建構出結構化的多屬性、多層環 (Multi-Attribute Range Query, MARQ) overlay，再配合雙向連結，除了解決 Chord 延展性不佳的問題，還可以提供多屬性、範圍性的搜尋，更加快搜尋的速度，以下分三個階段說明系統的架構。

(一)節點 ID 設定

Chord 在訂定節點 ID 並有沒特定的方式，節點 ID 僅給節點訂定編號，所以只能提供關鍵字搜尋。本研究利用屬性值設定節點 ID，讓節點 ID 反應節點屬性。節點 ID 分為屬性 ID 與亂數 ID 兩部份。在屬性 ID 部份，為了結合屬性的觀念，必須建立起節點 ID 與節點屬性值之間的關連性，所以在設定屬性 ID 時，先把屬性 ID 分成多組欄位，每個欄位對應一種屬性，然後將節點的每一種屬性值經過 hash 後所得到的結果，存在相對應的屬性欄位中，就可以得到一組與節點屬性值相關的屬性 ID。屬性值的 hash function 為：

$$f_a = (2^m \div (MAX_a - MIN_a)) \times (V_a - MIN_a) \text{。(1)}$$

為了避免不同節點擁有相同屬性值，所以在屬性 ID 之後，加上一組由亂數產生的 ID，即為此節點的節點 ID。以圖 1 為例，節點 ID 為 10 bits，其中屬性 ID 佔 6 bits，亂數 ID 佔 4 bits。在屬性 ID 中，前兩個 bits 表示 CPU 效能，中間兩個 bits 表示 CPU 數量，最後兩個 bits 表示記憶體大小，而節點 p 在 CPU 效能、CPU 數量、與記憶體大小的值分別為 2G、2、512M，經過 hash 後的結果分別為 01、10、01，將此結果存入對應的屬性欄位所產生的屬性 ID 為 011001，再加上一組亂數 ID 0111，所得到的 ID 011001 0111 即為節點 p 的節點 ID。因此，從節點 ID 就可以了解該節點的屬性，進一步達到屬性搜尋。

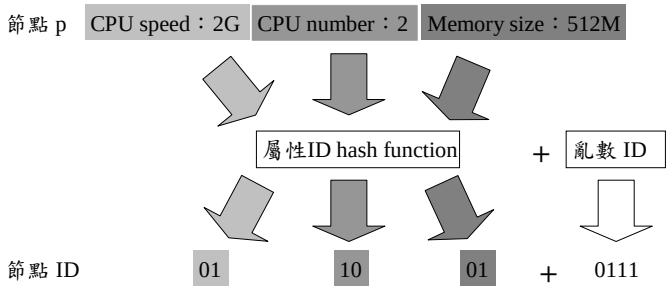

圖 1 Hashing function

(二) 建立 overlay

Overlay 的建構方式如下列演算法，主要是根據屬性 ID 而建立，設總節點數為 n，節點 p_i 表示第 i 個節點，節點 ID 分為 m 個屬性，每種屬性值由 b bits 表示，t_j 表示第 j 個屬性的屬性值，a_{ij} 表示節點 i 的第 j 個屬性值，Aj[]儲存第 j 個屬性環上的節點。

演算法 MARQ Overlay construction

Input：依節點 ID 循序排列的節點
Output：New overlay

```
Begin
    for j := 0 ~ m-1
        t_j := a_0j ;
    for i := 0 ~ n-1
        for j := 0 ~ m-1
            if   a_ij = t_j
                if   j = m-1
                    put p_i in A_m[];
            else
                for k := j ~ m-1
                    put p_i in A_j[];
                    DL_Chord(A_k+1[]);
                    t_k = a_ik ;
end
```

建立 MARQ overlay 可分為三個步驟：

步驟一：將節點根據屬性 ID 依序排列，在每個節點 P_i 的 finger table 內，加入節點 P_{i+1} 的連結，而最後一個節點 P_{n-1} 的 finger table 則加入第一個節點 P_0，形成一個環狀的 overlay。以屬性 ID 為 6 bits，分成 3 種屬性，每種屬性值為 2 bits 為例，經過步驟一所形成的環狀結構如圖 2 所示

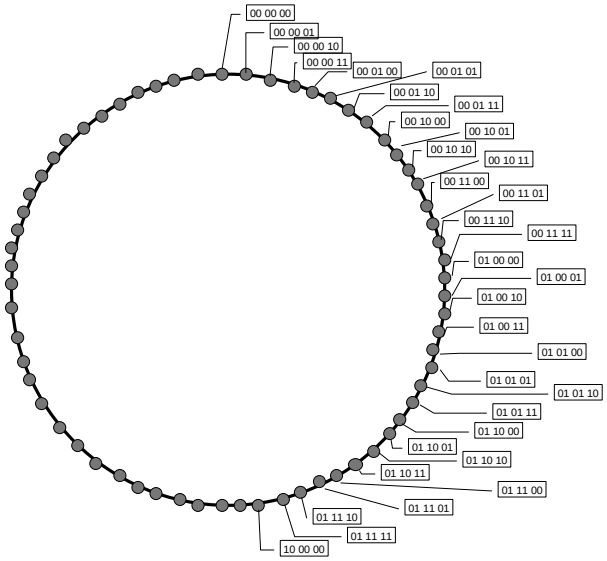

圖 2 三種屬性的 MARQ 環狀結構

步驟二：建立第一組屬性環，把第一組屬性值相同的節點設為同一個區塊，所以由步驟一所產生的環狀 overlay 可以分割成 2^b 個區塊，再將每個區塊的第一個節點用步驟一的方式建立出第一組屬性的屬性環(如圖 3)，除了第一組屬性的屬性環外，還會產生另外 2^b 個子環，而在同一子環中的屬性 ID，其第一組屬性值皆相同

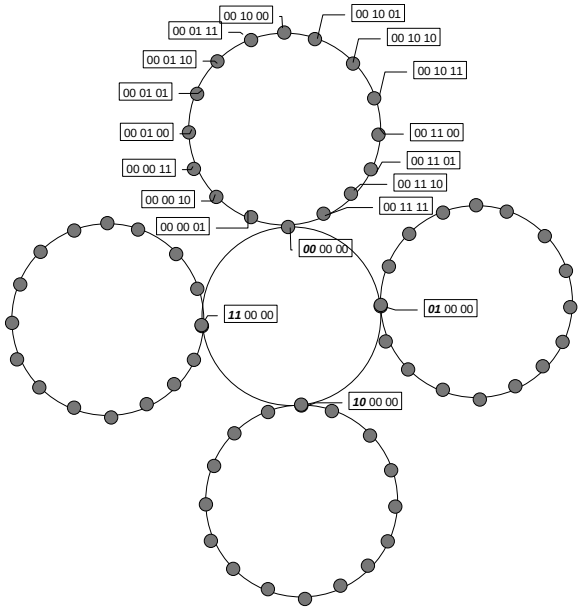

圖 3 第一組屬性的屬性環

步驟三：建立每個屬性的屬性環，在每個子環中，依序根據第二組、第三組屬性值，重覆步驟二，即可建立出 *m* 種屬性的屬性環，形成一個多屬性、多環的 overlay，如圖 4

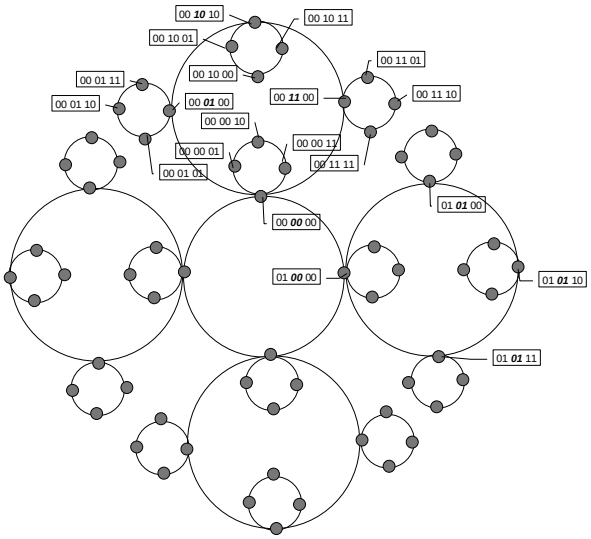

圖 4 多重屬性環

(三) 建立雙向連結

最後是建立雙向連結，如下列演算法。

演算法 DL_Chord (A[])

Input：儲存同屬性環上節點的陣列(A[])

Output：雙向連結的 Chord

Begin

for i := 0 ~ *A*[].*size* - 1

for j := 0 ~ *log*(*A*[].*size*)

put $A[(i+2^j)\%A[].size]$ in f_i ;

put p_i in $f_{A[(i+2^j)\%A[].size]}$;

A[]:=0 ;

end

傳統 Chord 所建立的 finger table 中，記錄了 log(*n*)個後繼節點。訊息只能單向地由節點往後繼節點傳送，而雙向連結則是當節點把後繼節點加入 finger table 中時，該後繼節點也會把此節點加入自己的 finger table，兩節點之間便建立了雙向的連結，可以互相傳訊。所以每個節點的 finger table 所記錄的節數，除了原來的 log(*n*)個後繼節

點，另外還會增加 log(*n*)-1 個祖先節點，finger table 內的記錄的連結數量為 2log(*n*)-1。因此，搜尋訊息可以同時往順時鐘(往後繼節點)與逆時鐘(往祖先節點)兩個方向傳送，傳送訊息時有更多的路徑可供選擇。因此，有效縮短了搜尋節點數、搜尋時間，還可以達到範圍搜尋的功能。

(四) 動態操作

在 P2P 網路中，時常發生節點加入或離開，此時，overlay 需要動態的調整，以保持整個 overlay 的完整性。

節點加入：欲加入的節點先經由 hash 得到屬性 ID，接著在 overlay 上搜尋相同屬性 ID 的節點。若存在，則在進入該節點的屬性環後，根據亂數 ID，加入到該屬性環內；若不存在，則建立自己的屬性環。

節點離開：當節點離開時，若節點不是某個屬性區塊中的第一個節點，則直接離開系統，其所屬屬性環內的節點更新 finger table 即可；若離開的節點是屬性區塊中的第一個節點，則由次一個屬性環上的第一個後繼節點，來繼承離開節點的位置。

四、資源搜尋

本研究以 Chord 為基礎，除了保留 Chord 在關鍵字搜尋上的快速、有效外，還可提供多屬性搜尋與範圍搜尋兩種功能。

(一) 關鍵字搜尋

Chord 所建立的結構化 overlay，可提供快速、有效的關鍵字搜尋，但由於 Chord 上的 finger table 只記錄了後繼節點，所以訊息只能順時鐘單向地往後繼節點傳送。而本研究採用了雙向連結，使得訊息不但能以後繼節點傳送，還能以逆時鐘的方向往祖先節點傳送，讓訊息在傳送時，有更多路徑可以選擇。因此，比傳統 Chord 使用更少的搜尋節點數、更短的時間，就能找到所需的資源。

(二) 多屬性搜尋

因節點 ID 的屬性 ID 是由節點的屬性值 hash 後所得到的，所以從屬性 ID 就可以知道節點的屬性。在搜尋時，先將使用者要搜尋的資源之屬性值 hash 後放入與屬性 ID 相同的欄位中，即可得到 query ID。將此 query ID 提交到第一屬性環上，在第一屬性環上搜尋第一個屬性相同的節點，找到後再進入該節點的第二屬性環，搜尋第二屬性值相同的節點，以此類推，即可找到所需要的資源。以圖 5 為例，使用者欲搜尋一個 CPU speed 為 2G，CPU 數量為 2，Memory size 為 2048M 的運算資源，而這三個屬性值經 hash 後分別為 01、01、11，放入與屬性 ID 相對應的欄位中，得到 query ID 為 010111，提交到第一屬性環上找到第一屬性相同的節點 010000，接著在節點 010000 的第二屬性環中搜尋第二屬性值相同的節點，所以找到了節點 010100，最後在節點 010100 的第三屬性環中搜尋第三屬性值相同的節點，即為滿足使用者需求的運算資源。

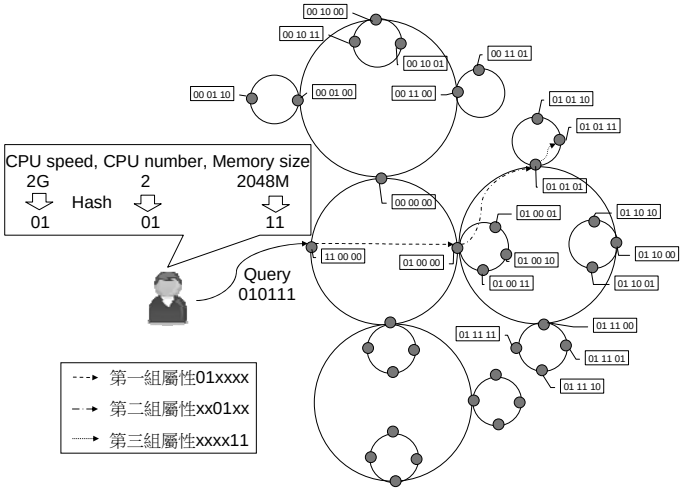

圖 5 多屬性搜尋

(三) 範圍搜尋

範圍搜尋可分為兩種，一種是搜尋部份屬性值相同的節點，另一種是搜尋屬性值介於某範圍之間的節點。

搜尋部份屬性值相同的節點，使用者可指定部份的屬性值，而其它屬性值則不限定，其搜尋方式與多屬性搜尋類似。首先 query ID 提交到第一屬性環，在第一屬性環上找到第一屬性值與 query ID 的第一組屬性值相同的節點，接著在該節的第二屬性環上搜尋第二屬性值與 query ID

的第二組屬性值相同的節點，依此類推，直到找到屬性值與指定屬性值相同的節點後，該節點內的其它屬性環上的節點，即為目標節點。以圖 6 為例，query ID 為 0110xx，先在第一屬性環上找到節點 010001 其第一屬性值相同，接著在節點 010001 的第二屬性環上搜尋第二屬性為 10 的節點即節點 011000，最後由節點開始做 one-hop 搜尋，直到搜尋訊息回到節點 011000，所經過的節點即為所求。

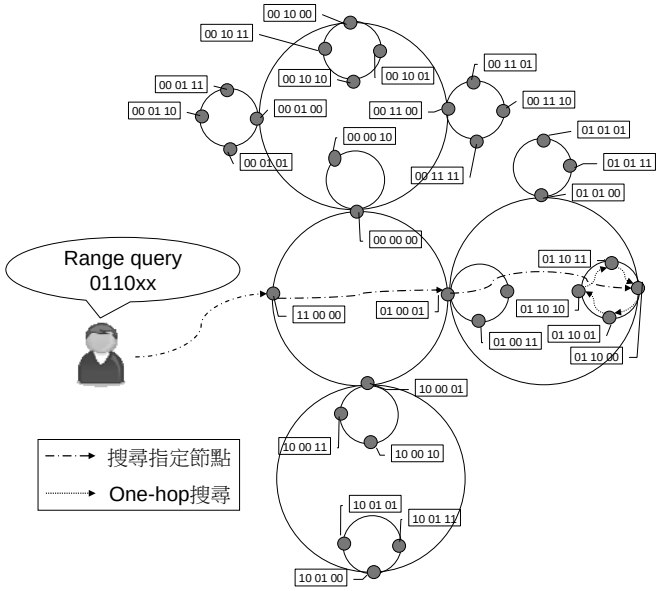

圖 6 搜尋部份屬性值節點

若搜尋屬性值介於某範圍之間的節點時，則採用下列 Range query 策略。在執行此類範圍搜尋時，先同時送出兩個搜尋訊息，分別找出符合所求範圍之最大值與最小值的節點，找到兩端節點後，再由最大值的節點逆時鐘(往祖先節點)做 one-hop 搜尋，而最小值的節點則由順時鐘(往後繼節點)做 one-hop 搜尋，直到兩個搜尋訊息碰撞才停止，而介於兩端節點之間的所有節點，其屬性值都在所求的範圍之內。以圖 7 為例，搜尋的範圍介於 000100 到 001110 之間，在提出 query 後，分別找出上限節點(節點 001110)與下限節點(節點 000100)後，再用 one-hop 搜尋找出兩節點之間的所有節點。

演算法 Range query (lower bound, higher bound)

Input：範圍搜尋的下限與上限(lower bound, higher bound)

Output：指定範圍內的節點

begin

divide the range query into lower bound query and higher bound query :

use Chord-based routing protocol to find the lower bound node p_l and the higher bound node p_h ;

while lower bound query not collide higher bound query

for p_l, send the query to the node which node ID is one hop great than p_l in the finger table ;

return p_l;

for p_h, send the query to the node which node ID is one hop less than p_h in the finger table ;

return p_h;

end

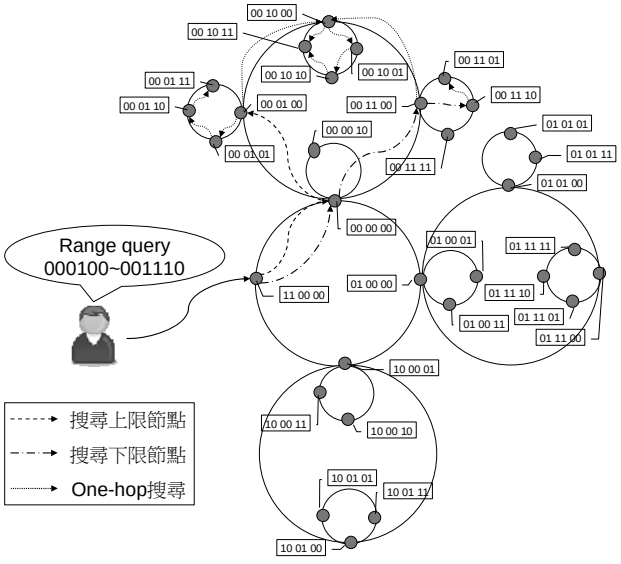

圖 7 搜尋範圍屬性

五、實驗設計與結果分析

(一) 實驗設計

本實驗利用 OverSim 模擬 P2P 的網路環境。OverSim 是一套 overlay 網路模擬架構，可以用來模擬各種的 overlay，裡面也有多種常見 P2P 網路的模組，如結構化的 Chord、Kademlia、Pastry 等，和非結構化的 P2P 協定，方便使用者架構 P2P 網路與數據的搜集、分析等等，圖 8 為本研究所提出之 overlay 利用 OverSim 所建立出之模式。

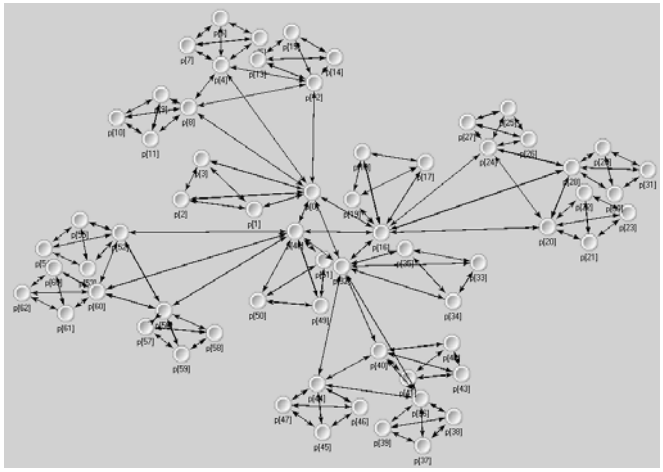

圖 8 OMNet++上的 MARQ 模擬結構圖

由於本實驗主旨在探討本研究所提出之 overlay 與傳統 Chord 在搜尋效能上的比較，所以為縮短模擬時間，在環境參數只簡單設定訊息傳遞時，節點與節點之間會有隨機 0.1 至 1 毫秒的延遲，網路頻寬為 1Mbps。比較方法為在不同的節點數量時，隨機取出三個節點做為目標節點，分別記錄搜尋訊息由其它節點傳送目標節點所花費的平均時間與節點數，如圖 9 為 Chord 在 128 個節點數時的平均搜尋時間。本研究所提出之 overlay 在不同節點數量時，屬性數與每種屬性的屬性值分配如表 1。

表 1 屬性值分配表

節點數	64	128	256	512	1024
屬性數	3	3	3	3	3
屬性值(bits)	2,2,2	2,2,3	2,3,3	3,3,3	3,3,4

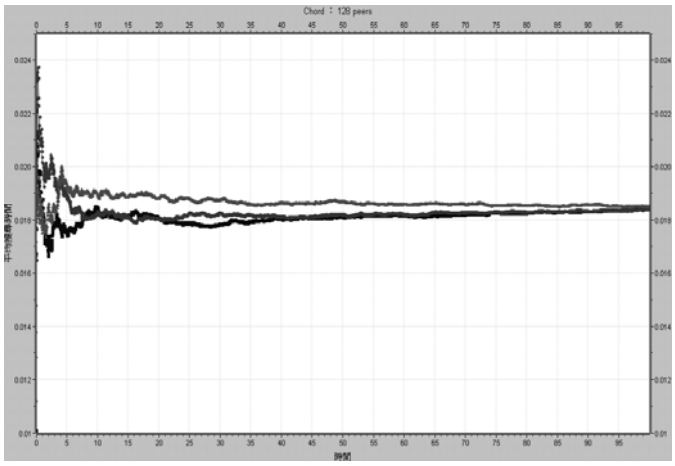

圖 9 . 128 個節點數時 Chord 的平均搜尋時間

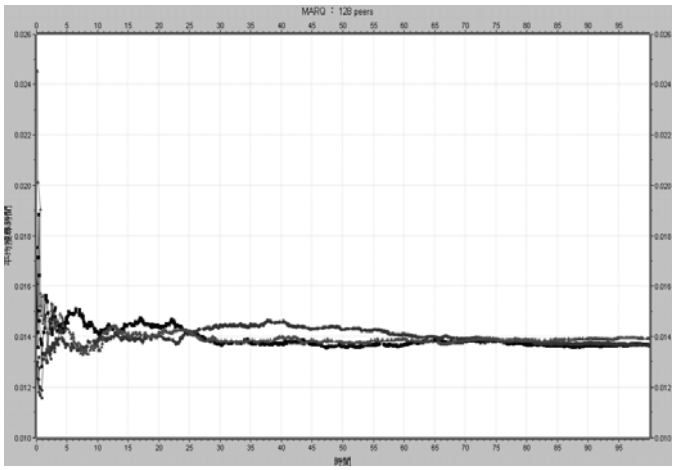

圖 10 . 128 個節點數時 MARQ 的平均搜尋時間

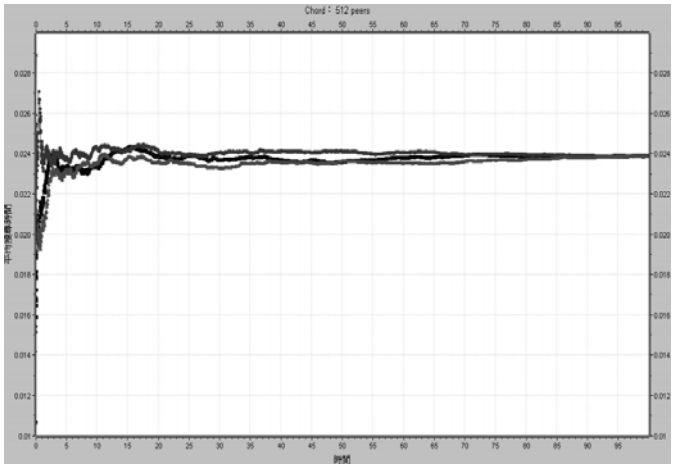

圖 11 . 512 個節點時 Chord 的平均搜尋時間

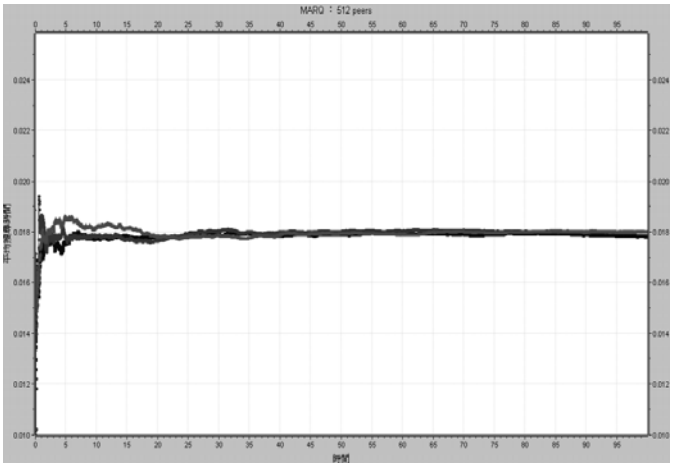

圖 12 . 512 個節點時 MARQ 的平均搜尋時間

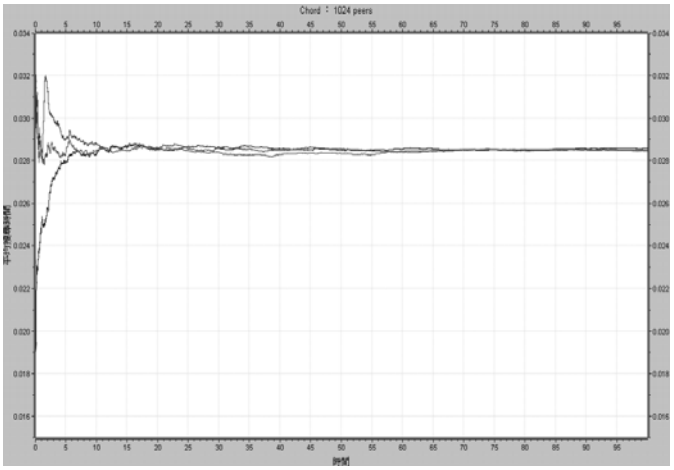

圖 13 . 1024 個節點時 Chord 的平均搜尋時間

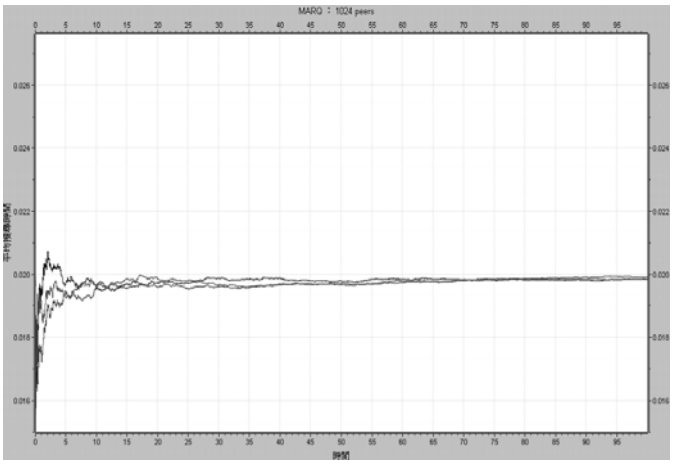

圖 14 . 1024 個節點時 MARQ 的平均搜尋時間

(二)結果分析

圖 9 至圖 14 為 Chord 與 MARQ overlay 分別在 128、512、1024 個節點時之平均搜尋時間。由圖顯示，在不同的節點數量時，MARQ overlay 只需較少的的搜尋時間，就能搜尋到所需的節點，且能在較短的時間內就能達到最短的搜尋時間。由圖 15 可以看出，MARQ overlay 的平均搜尋節點數比 Chord 少，這是因為 MARQ overlay 使用了雙向連結，讓搜尋訊息有更多的路徑可以選擇，因為搜尋節點數變少了，所以平均搜尋時間也隨著降低，由圖 16 可以看出，且隨著節點數量變多，兩者的差距會越明顯。

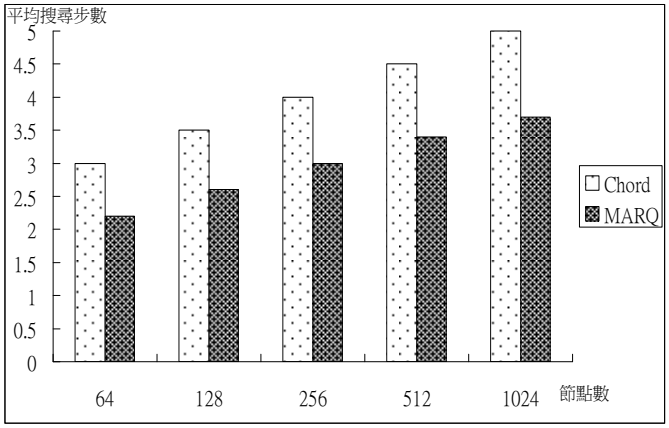

圖 15 平均搜尋節點數

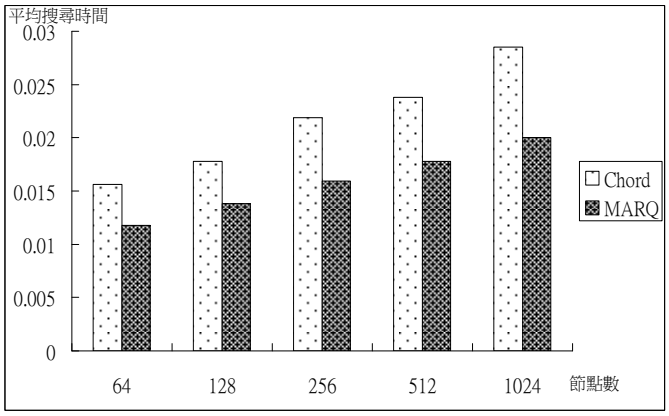

圖 16 平均搜尋時間

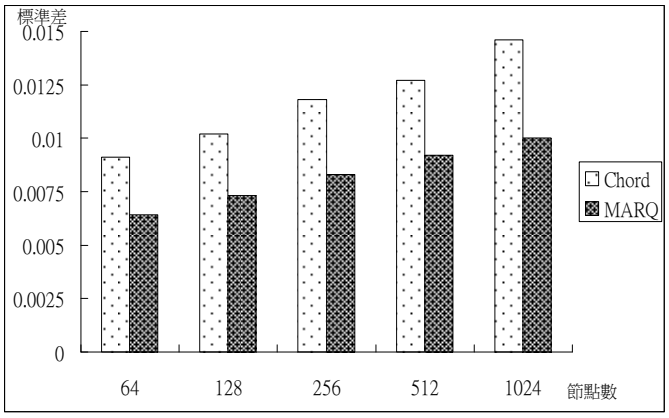

圖 17 標準差

雖然本研究使用雙向連結可有效的降低搜尋時間與搜尋節點數，但節點相對的必須在 finger table 中記錄更多的節點。假設 k 為屬性 ID 的 bit 數，若 Chord 使用雙向連結，則總連結總數為：

$$2(k-1)\times 2^k \text{。} \tag{2}$$

未使用雙向連結的總連結數為：

$$k\times 2^k \text{，} \tag{3}$$

對節點而言，使用雙向連結時，finger table 內需要記錄的節點數量比原來多了將近兩倍，但由於本研究所提出之 overlay 將節點分割成多個屬性環，而每個屬性環最多由 2^b 所組成，若有 m 種屬性，且每種屬性的屬性值由 b 個 bits 表示，即 $k=m\times b$，則使用雙向連結的 MARQ overlay 其總連結數為：

$$\sum_{i=0}^{m-1}[(2b-1)\times 2^b]\times 2^{ib} \text{。} \tag{4}$$

因此，使用雙向連結的 overlay 其總連結數比未使用雙向連結的 Chord 還少，且搜尋的效能優於 Chord

圖 17 為搜尋時的標準差，可看出當節點數量越多時，Chord 的標準差也越大，表示若發出搜尋訊息的節點與目標節點越遠，所需的搜尋時間越多。而 MARQ overlay，由於採用了多層屬性環的架構，所以當節點數量越多時，可有效降低搜尋訊息傳送到目標所需的時間。

六、結論及未來展望

本研究以 Chord 為基礎，建構出多屬性、多環的 MARQ overlay，每個屬性環上仍保有 Chord 的特點，能快速、有效的搜尋資源；且因為多環的結構，除了降低節點加入或離開時，維持 overlay 的花費，使用雙向連結後的總連結數還比 Chord 少，且可更進一步有效地降低平均搜尋時間與平均搜尋節點數。亦因將節點的屬性值反應在其節點 ID 上，使系統除了原來的關鍵字搜尋外，還可以提供多屬性搜尋、及範圍性搜尋的功能。

然而，MARQ overlay 若直接指定中間或後面的屬性值時，需產生較多的搜尋訊息，才能完成範圍搜尋。在未來研究中將會嘗試解決此問題，以最少的訊息量達到相同的搜尋效果。而本研究之初步實驗只與 Chord 進行搜尋效能上的比較，而未進行關於多屬性搜尋及範圍性搜尋的實驗，未來亦將針對多屬性搜尋及範圍搜尋進行效能測試。

七、致謝

This study was sponsored by the National Science Council, Taiwan, Republic of China under contract numbers: NSC 97-2221-E-142-001-MY3 and NSC 97-3114-E-007-001-.

八、參考文獻

[1] Chunming Hu, Yanmin Zhu, Jinpeng Huai,et al. “S-Club:An Overlay-based Efficient Service Discovery Mechanism in CROWN Grid,” Proceedings of 2005 IEEE International Conference on e-Business Engineering.2005:441-448.

[2] Gang Chen, Chor Ping Low and Zhonghua

Yang. “Enhancing Search Performance in Unstructured P2P Networks Based on Users’ Common Interest,” IEEE Transactions on Parallel and Distributed Systems, 19(6) :821-835 , 2008.

[3] Haiying Shen and et al. “PIRD: P2P-based Intelligent Resource Discovery in Internet-based Distributed Systems,” In Proc. of ICDCS, 2008.

[4] Ibrahim Al-Oqily and Ahmed Karmouch. “SORD: A Fault-Resilient Service Overlay for MediaPort Resource Discovery,” IEEE Transactions on Parallel and Distributed Systems, 20(8) :1112-1124 , 2008.

[5] James Salter and Nick Antonopoulos, “An optimized two-tier P2P architecture for contextualized keyword searches,” In Future Generation Computer Systems Journal, vol. 23, no. 2, 2007.

[6] Jiang Song, Guo Lei, Zhang Xiaodong, and Wang Haodong. “LightFlood : Minimizing redundant messages and maximizing scope of peer-to-peer search,” IEEE Transactions on Parallel and Distributed Systems, 19(5) :601-614, 2008.

[7] OMNeT++, "OMNeT++ Discrete Event Simulation System", http://www.omnetpp.org.

[8] OverSim, "OverSim: The Overlay Simulation Framework", http://www.oversim.org.

[9] Ronaldo A Ferreira, Mehmet Koyuturk, Suresh Jagannathan,et al. “Semantic indexing in structured peer-to-peer networks,” In J. Parallel Distrib. Journal, vol. 68, 2008.

[10] Simon Rieche, Bui The Vinh, Klaus Wehrle. “Range Queries and Load Balancing in a Hierarchically Structured P2P System,”

[11] Stephanos, Routsellis-Theotokis and Diomidis Spinellis. “A Survey of Peer-to-Peer Content Distribution Technologies," Comm. ACM 36, 4, 335–371.

[12] Yi-Hsiang Lin, “SARIDS：A Self-Adaptive Resource Index and Discovery System,” Master thesis, National Tsing-Hua University, 2009.

[13] Zeinalipour-Yazti Demetrios, Kalogeraki Vana and Gunopulos Dimitrios. “pFusion: A P2P Architecture for Internet-Scale Content-Based Search and Retrieva,” IEEE Trans-actions on Parallel and Distributed Systems, 18(6):804-817.