國 立 交 通 大 學
資 訊 工 程 學 系
碩 士 論 文

DLC：可擴展式點對點查詢與資料分享架構

DLC：A Scalable Peer-to-peer Lookup Service and Data Sharing Architecture

指導教授：蔡文能　教授
研 究 生：陳積弘

中 華 民 國 九 十 四 年 七 月

DLC：可擴展式點對點查詢與資料傳輸架構

DLC：A Scalable Peer-to-peer Lookup Service and Data Sharing Architecture

指導教授：蔡文能　　Advisors：Wen-Nung Tsai

研 究 生：陳積弘　　Student：Chi-Hung Chen

國 立 交 通 大 學

資 訊 工 程 研 究 所

碩 士 論 文

A Thesis Submitted to
Institute of Computer Science and Information Engineering
College of Electrical Engineering and Computer Science
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of Master
in
Computer Science and Information Engineering

July 2004

Hsinchu, Taiwan, Republic of China

中 華 民 國 九 十 四 年 七 月

可擴展式點對點查詢與資料分享架構

學生：陳積弘　　指導教授：蔡文能 教授

國立交通大學資訊工程學系（研究所）碩士班

摘　　要

近年來，隨著用戶端的設備日漸強悍，點對點的應用軟體也日漸普及，目前已有許多不同種類的點對點架構，然而大部分的架構都著重於資料查詢或者如何維護與其他用戶之間的溝通。往往沒有考慮到部分用戶端因被選爲熱點(中繼點)而造成該用戶系統網路壅塞及效能低落。

本論文提出的 DLC 是修改自既有點對點架構 Chord，試圖解決上述問題。另外，我們額外在查詢與傳輸著墨，試圖提升有效查詢、有效傳輸、減少網路資源浪費特別是針對熱門資料。

A Scalable Peer-to-peer Lookup Service and Data Sharing Architecture

student：Chi-Hung Chen **Advisors：Dr. Wen-Nung Tsai**

Institute of Computer Science and Information Engineering

National Chiao Tung University

ABSTRACT

In recent years, as the end users' equipments get more and more powerful, the peer-to-peer applications become more and more popular. There are many different kinds of peer-to-peer architectures. However, most of them only focus on how to search data and/or how to maintain neighbor relations. Unfortunately, some hosts selected to be hot points (via points) may result in low performance due to heavy overhead.

In order to solve these problems, we propose a new P2P architecture, called DLC, that base on Chord. In DLC, we pay more attention to the transmission and search over this logical topology. Experimental results show that our approach can improve the efficiency of query and data transmission and thus reduce the network resources consumption, especially for popular data.

致　　謝

在兩年的努力之下，終於完成了我的畢業論文，中間經過了許多的困難與問題，也受到許多人的幫助，在此一一感謝。首先要感謝的是我的指導教授-蔡文能教授，在碩士班的兩年，他給我許多方面的指導，讓我受益良多，也成長了不少。再者，感謝父母兄姐、奶奶給我一個平穩的環境，讓我能夠專心學習。接著要感謝的是實驗室的學長姐、同學們，大家一起工作、學習，互相幫忙，讓我有個難忘的碩士回憶。需要感謝的人實在太多了，如教會朋友、大學好友...等，在我沮喪或難過時，他們都能即時給我幫助與支持，謝謝他們。最後，感謝上帝與我同在。

目　　錄

摘　　要 i
ABSTRACT ii
致　　謝 iii
目　　錄 iv
表　目　錄 vi
圖　目　錄 vii
第一章 緒論 1
1.1 簡介 1
1.2 動機與目的 3
1.3 論文架構 3
第二章 背景知識 4
2.1 點對點應用(Peer to Peer Application) 4
2.1.1 檔案下載 5
2.1.2 網路即時通訊 6
2.2 點對點傳輸架構世代 7
2.2.1 主從式傳輸（Client and Server）架構 7
2.2.2 第一代點對點傳輸架構 8
2.2.3 第二代點對點傳輸架構 9
2.2.4 第三代點對點傳輸架構 11
2.2.5 其他點對點網路傳輸架構 12
KaZaA 12
Bittorent 13
2.3 多址傳輸（Multicast）與路由（Routing） 14
2.3.1 IP Multicast 14
2.3.2 應用層多址傳輸（Application Layer Multicast） 15
2.3.3 IP Routing 16
2.3.4 應用層路由（Application Layer Routing） 16
2.4 地域(Locality) 17
2.4.1 網路位置定位(IP Locality) 18
2.4.2 RTT技術 － 地標(Landmark) 18
2.4.3 代理(Proxy) 19
第三章 相關研究 20
3.1 Routing Index 20
3.2 CAN 21
3.3 Chord 24
3.4 Pastry 25
3.5 Kademlia 27
第四章 系統架構 29
4.1 系統概觀(System Overview) 29
4.2 設計方法與假設 31

4.2.1 雙向搜尋(Double Direction Search) 31
4.2.2 主動地域代理(Active Locality Proxy) 33
4.2.3 BT追蹤者傳輸(BT Tracker Transmission) 34
4.3 用戶端軟體架構 35
4.3.1 傳輸管理元件(Transmission Manager) 35
4.3.2 路由管理元件(Routing Manager) 36
4.3.3 資料追蹤管理元件(Tracker Manager) 36
4.3.4 分享檔案管理元件(Share Data Manager) 37
4.4 主要程序訊息 38
4.4.1 查詢(Look Up) 38
4.4.2 更新(Update) 39
4.4.3 分享(Share) 40
4.4.4 加入(Join) 41
4.4.5 離開(Depart) 42
第五章 模擬結果 43
5.1 P2PNS簡介 43
5.2 評量指標(Criteria) 47
5.3 系統效能 (Performance Evaluation) 48
5.2.1 搜尋相關 49
5.2.2 傳輸相關 50
5.2.3 負載相關 51
第六章 結論 52
6.1 討論 52
6.2 結論 53
6.3 未來工作 54
參考文獻 55

表　　目　　錄

表格 1 台灣常見點對點軟體比較 5
表格 2 台灣常見及時通訊軟體比較表 6
表格 3 Chord與DLC的路由表區間 31
表格 4 CAN、Torus CAN、Chord、Double Chord搜尋跳躍比較 32
表格 5 程式碼詞彙表 38
表格 6 分享程序訊息 41
表格 7 綜合比較 52

圖　　目　　錄

圖表 1 雜湊邏輯網路 2
圖表 2 邏輯佈設與實體網路 4
圖表 3 搜尋介面(Enule搜尋介面) 5
圖表 4 VOIP演進 6
圖表 5 主從傳輸架構 8
圖表 6 第一代點對點傳輸架構 9
圖表 7 第二代點對點網路傳輸架構 9
圖表 8 Gnutella搜尋邏輯 10
圖表 9 Gnutella 的搜尋影響 10
圖表 10 第三代點對點網路傳輸架構 11
圖表 11 KaZaA網路架構 12
圖表 12 BT傳輸架構 13
圖表 13 Unicast與Multicast 15
圖表 14 Application Layer Multicast 15
圖表 15 IP Routing 16
圖表 16 Skype 封包路由 16
圖表 17 無Land Mark 的最差情形 17
圖表 18 具Land Mark的理想情形 17
圖表 19 RTT技術 19
圖表 20 Proxy架構 19
圖表 21 RI 架構 20
圖表 22 RI 各用戶端資料表的關係 21
圖表 23 CAN的搜尋邏輯 22
圖表 24 CAN維度與路徑長度關係 23
圖表 25 CAN網路實體個數與路徑長度關係 23
圖表 26 CAN的實體個數、維度與路徑長度關係 23
圖表 27 Chord Finger Table 24
圖表 28 Pastry搜尋 25
圖表 29 Pastry 各用戶端要維護的表 26
圖表 30 Pastry的效能測試 26
圖表 31 Kademlia提出Chord的缺點 27
圖表 32 相對子樹 27
圖表 33 Kademlia的搜尋 28
圖表 34 DLC實體網路架構 29
圖表 35 DLC網路邏輯架構圖 30
圖表 36 DLC與Chord比較(1) 32
圖表 37 DLC與Chord比較(2) 32
圖表 38 DLC與Chord比較(3) 32

圖表 39 用戶端編號 33
圖表 40 BT追蹤者傳輸 34
圖表 41 DLC用戶軟體架構 35
圖表 42 DLC尋找後繼者(find_successor) 39
圖表 43 正常更新 40
圖表 44 非正常更新 40
圖表 45 加入 41
圖表 46 正常離開 42
圖表 47 P2P Network Framework 44
圖表 48 Action Generator 45
圖表 49 P2PNS系統概觀 46
圖表 50 搜尋成功率 49
圖表 51 搜尋路徑長度 49
圖表 52 傳輸成功率 50
圖表 53 傳輸百分比 50
圖表 54 路由器傳輸負載 51
圖表 55 用戶端搜尋負載 51
圖表 56 階層式點對點傳輸架構 54

第一章 緒論

1990 年代網際網路興起，網路使用者數量急遽上升，越來越多應用都是透過網路來進行。到了 1990 末期，隨著用戶端設備日漸強，以及網路頻寬加大，點對點應用儼然成爲最熱門的網路傳輸方式。雖然目前已有許多不同種類點對點網路架構，但是主要基礎架構多出於集中索引(Index Server)、分散式服務(Distribute Service)、分散式雜湊表(DHT，Distribute Hash Table)等三種，各有各的優缺點，但尚未有一個最佳的架構。

現行的架構多著重於資料查詢或者與其他用戶端之間的溝通維護。但是往往沒有考慮到部分用戶因被選爲熱點(via point，中繼點)而造成該用戶端網路壅塞及效能低落，更嚴重地，萬一該用戶故障，則和該用戶相關的資訊不僅全部斷除，更需要花上許多回復成本。另外，單靠路由器等硬體設備無法有效降低網路頻寬，然而今日用戶設備多有相當高水準，足以輔助封包繞送。如何合理且有效地利用用戶端資源，使同一點對點架構下的所有用戶端，除了保有一般點對點網路架構的優點之外，更能避免熱點、降低整體網路成本都是值得探討的問題。

本論文嘗試修改既有點對點架構 Chord，加上一些理論設計與新思維，提出一個新架構 DLC(Double direction Locality Chord)，希冀其可在不增加太多額外負載的限制下，合理解決上述問題。

1.1 簡介

雜湊(Hash)是龐大資料下最快速的搜尋方式。其主要由兩個部分組成：一個良好的雜湊函式(Hash Function)以及一塊分割並編號完整的邏輯位置空間。資料置入儲存空間時，並非任意放置，而是先利用資料特徵值透過雜湊函式計算出對應的邏輯位置編號，然後將資料置放在編號同計算結果的位置空間。所以搜尋時，只要算出邏輯位置編號即可快速取得資訊。由於雜湊效能相當顯著，其觀念亦可應用在網路搜尋上。

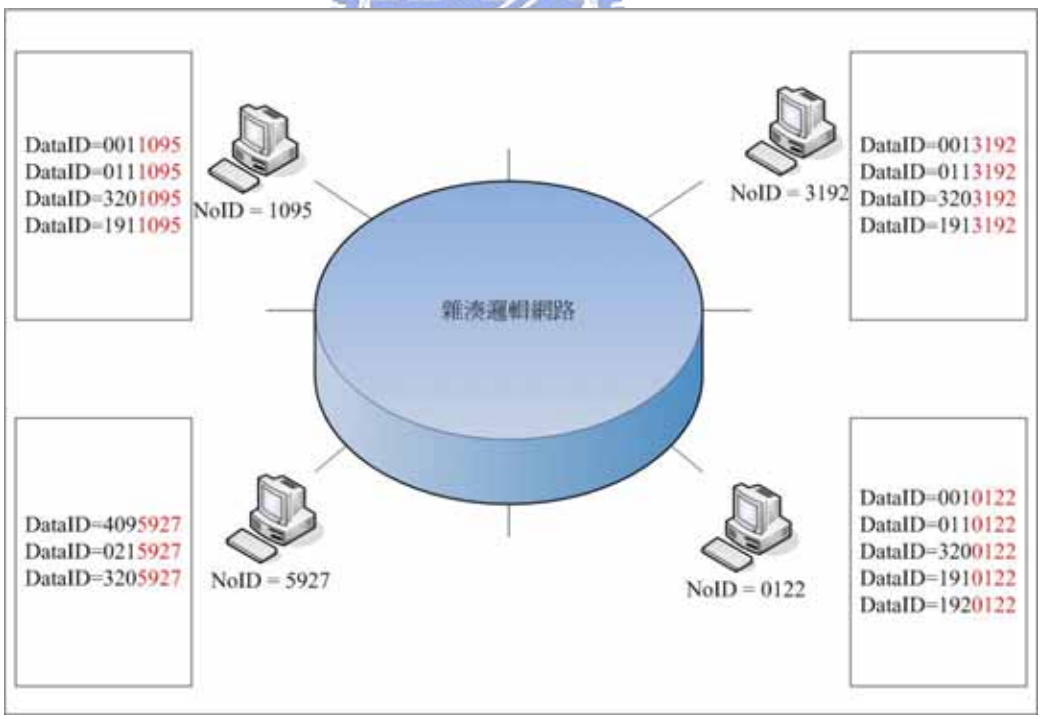
近期提出的第三代點對點網路架構，即利用雜湊概念來定位用戶端及資料的邏輯位

置。一般應用上，用戶端定位通常會利用 IP、Port 等做爲特徵值，而資料通常以資料的屬性如名稱等做爲特徵值。

用戶端編號 = Hash(IP, Port) ;

資料編號 = Hash(Name) ;

當用戶端的用戶端編號算出後，經由一個已在該邏輯網路架構下的用戶端協助，進入該邏輯網路架構。進入之後，用戶端將取得鄰居資訊，並藉由一連串的動作觸發相關鄰居，更新相關鄰居資訊。而當資料的資料編號算出後，該資料的媒體資料(Meta Data)將被應對到某用戶端上，而該用戶端的編號則爲當時最接近資料編號的一個。所以搜尋時，只要算出資料編號即可找到該資料的媒體資料，進而取得資料實際位置資訊。另外，資料總數往往遠多餘用戶端總數，所以一般架構的資料編號空間都會大於用戶端編號空間，一般皆採取餘數的概念來解決此問題，以圖表 1 爲例，編號爲 3192 的用戶端，會負責所有資料編號餘數爲 3192 的資料。



圖表 1 雜湊邏輯網路

1.2 動機與目的

以雜湊爲基礎的點對點架構，的確有相當多好處。然而一般的應用卻僅限於資料查詢輔助，卻很少注重檔案傳輸。一般來說，擁有資料的用戶端，常常沒有一定的規則，所以要求傳輸資料時，往往都是要找到原有資料的用戶端，否則就是利用用戶端的歷史資訊，來找尋其他擁有資料實體的用戶端。

本論文認爲此種架構除了查詢之外，更可輔助資料傳輸，只要再加入類似代理、Bittorrent 的共享概念，並利用應用層來做資料傳遞輔助，應該有助於實體傳遞。再者由於提供資料多的用戶端，往往成爲其他用戶端搜尋或傳輸的目標，本論文對此也提出一些概念與想法，以減輕其負載。

1.3 論文架構

論文將以點對點傳輸架構演進的順序，說明其改進的源由以及優缺點，同時講述相關知識。另外，再提出其他既有架構並比較之，接著提出 DLC 架構與其設計概念，最終以模擬來驗證。本論文的整體組織如下：

『第二章 背景知識』介紹本論文中應用的技術及概念。

『第三章 相關研究』探討現行點對點架構的概念及優缺點。

『第四章 系統架構』進一步說明 DLC 的規劃與細部內容。

『第五章 模擬結果』以模擬實驗來驗證 DLC 效能。

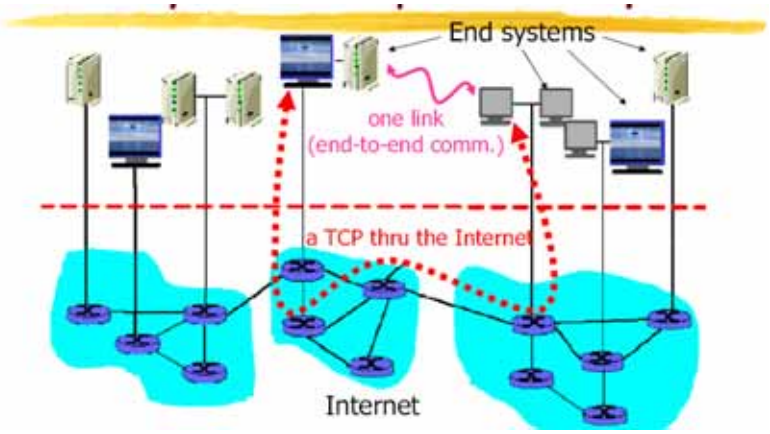
『第六章 結論』對本論文做結論、比較，並提出未來方向。

第二章 背景知識

在此章節將一一介紹點對點應用的實例、分類以及相關知識。2.1 介紹實際點對點的應用包含檔案下載以及最熱門的即時通訊軟體。2.2 將點對點架構依照其發展的時間與特性作分類說明。2.3 介紹應用層多址傳送(Application layer Multicast)及應用層路由(Application layer routing)的概念。2.4 講述地域(Locality)技術，如何做到將邏輯網路接近實體網路，以便輔助資料的傳輸。

2.1 點對點應用(Peer to Peer Application)

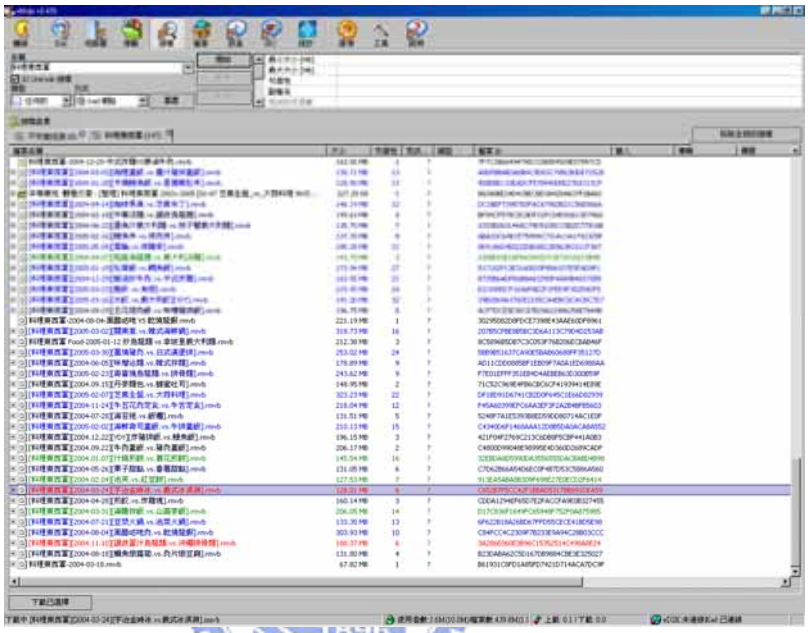
點對點應用主要概念是在現有實體網路上做邏輯佈設，並非重新建構新實體網路，用戶端不管選用哪種邏輯架構，最終資料傳輸與訊息傳送，還是藉由實體網路的 IP 傳輸。兩個用戶端之間各種傳輸，不管是直接傳輸或透過其他用戶端的間接傳輸，基本上都是用戶端兩兩互相溝通傳遞。然而這些邏輯架構的優劣比較就在於佈設上是否穩固、有效率及所提供服務的多寡。在此小節，我們將介紹目前台灣最熱門檔案下載和即時通訊軟體。



圖表 2 邏輯佈設與實體網路

2.1.1 檔案下載

Emule、Edonkey、Bittorrent 是目前台灣最熱門的點對點檔案下載軟體，WinMax、Kuro、Ezpeer 可算是目前台灣最熱門的點對點音樂下載軟體。這些軟體的主要目的就是做到資源共享。



圖表 3 搜尋介面(Enule 搜尋介面)

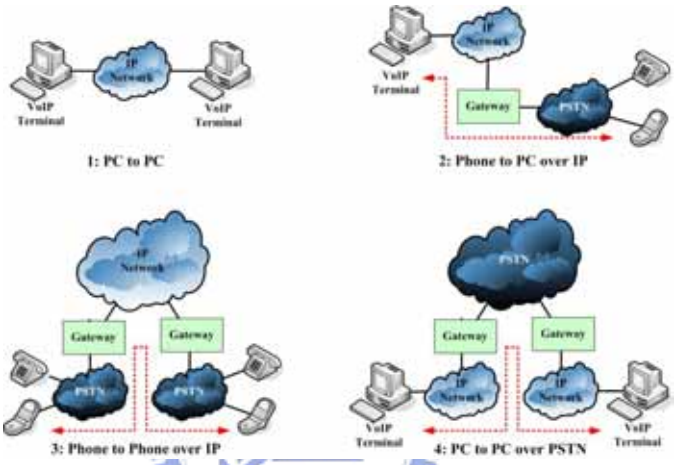
搜尋可謂此類軟體的主要服務之一，除了 Bittorrent 必須由使用者自行尋找所求檔案的種子(SEED)之外，其他軟體皆提供搜尋功能與介面(如圖表 3)，使用者可輕易找出所求檔案的來源用戶端。然而目前尚未有最佳搜尋架構，所以各軟體各有各的優缺點。搜尋廣度、搜尋速度、檔案可取得性、搜尋頻寬等都是評比關鍵。而整體架構效能，除了和軟體本身有關，使用者的參與行爲也是影響效能的重要因數。

	Emule	Edonkey	Bittorrent	KaZaA	WinMax	Kuro	Ezpeer
架構世代	1/3	1	N	3	1	1	1
搜尋功能	有	有	無-SEED	有	有	有	有
加密/格式	Y-RSA	Y-Key handshake	Y-FBT2	Y-RSA	Y - ABELSoft	Y	Y
通過 NAT	不可	不可	不可	可	不可	不可	不可

表格 1 台灣常見點對點軟體比較

2.1.2 網路即時通訊

即時通訊可謂現今最熱門的網路應用軟體，從早期的 ICQ 到 MSN、Yahoo Messenger、Skype。基本上都是建構在點對點傳輸之上。尤其後來紛紛加入 VOIP 功能，使得點對點傳輸特性更爲明顯。即時通訊的概念相當容易，藉由設備將用戶端發出的訊息包裝成封包，經由媒體傳輸到對方接收設備，可處理這類工作的設備不再限於電腦，目前已有商業販售的其他硬體設備。概括分類這些 VOIP 應用，依演進過程可分爲 PC to PC、PC to Phone、Phone to Phone 等三種，以圖表 4 說明這三種系統的整體架構。



圖表 4 VOIP 演進

這些 VOIP 應用中最引人注目的應屬 Skype，因爲其音質最佳。除了採用適當的編碼 GIPS iSAC and iLBC codecs[27]之外，其封包繞送也有別於一般 VOIP 軟體的直接點對點傳輸。雖然其架構到目前爲止尚未正式公開，但是分析指出，其採用了特殊的應用層路由(Application layer routing)來輔助封包傳送[7]。藉充分利用用戶端網路設備來增加傳輸的穩定性以求提供更好的通話品質，然而這類作法在實際產品 KaZaA -Lite 中證實是可行的[23]。表格 2 爲針對目前最熱門的即時通訊軟體所做的簡單優劣比較。

	ICQ	Yahoo Messenger	MSN	Skype
通話品質	可	可	可	優
視訊	有	有	有	無
通過 NAT	不可	不可	不可	可

表格 2 台灣常見及時通訊軟體比較表

2.2 點對點傳輸架構世代

本節以演進時間作爲對點對點邏輯架構的分類：主從架構(Client and Server)、第一代、第二代、第三代及其他架構。此外各小節也將講述各個架構的基本概念和限制，並提出優缺點。

2.2.1 主從式傳輸（Client and Server）架構

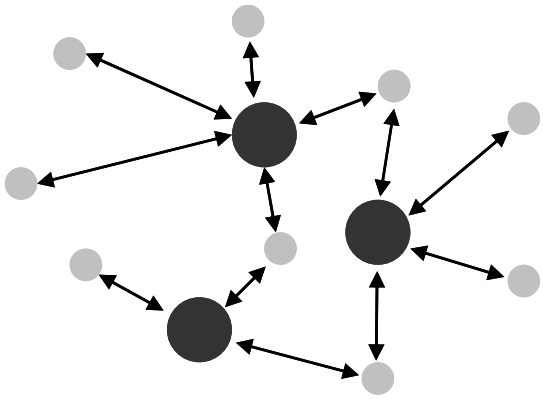
主從架構是最早透過網路傳輸的方式。由用戶端(Client)和伺服器(Server)組成。雙方必須以兩方皆認同的通訊埠(Port)和通訊協定(Protocol)來溝通。常見應用首推檔案傳輸協定(FTP)。加入主從式傳輸架構用戶端只需要知道伺服器網路位置、通訊埠與通訊協定即可。用戶端搜尋資料時，直接向伺服器發出詢問即可。

歸納此一架構的優點如下：

- 通訊協定簡單。
- 一般擁有較穩定的傳輸速度。
- 容易確保下載檔案的完整性及統一性。
- 伺服器可有效控管檔案的讀取權限。

歸納此一架構的缺點如下：

- 有固定的伺服器，容易成爲攻擊的目標，一旦伺服器被攻破或者故障，則資訊的來源隨即停止。
- 分享服務只能由伺服器端提供，用戶端要分享資源時，必須經由伺服器同意，然後上傳到伺服器。造成檔案分享的限制。
- 用戶端必須知道伺服器等相關資訊。
- 伺服器的負載重，除了提供搜尋還要負責傳輸資料。



圖表 5 主從傳輸架構

2.2.2 第一代點對點傳輸架構

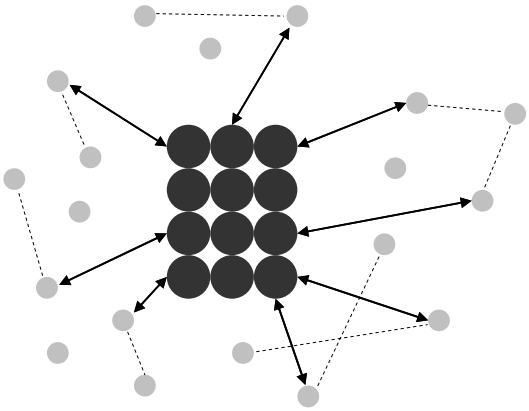
簡單地說，就是具有索引伺服器(Index Server)輔助檔案分享架構。主要是由索引伺服器和一般的點(Peer)組成，點即爲用戶端。索引伺服器顧名思義，就是伺服器上擁有大型集中式索引，其中記錄相連點有哪些資源可供分享。加入架構的方法很簡單，如同主從式架構，點只需知道索引伺服器網路資訊即可。當點搜尋資料時，直接向索引伺服器詢問，而索引伺服器則回傳擁有所需資料的用戶端資訊。所需資料的實體來源是回傳中的部分用戶端，並非索引伺服器。這也是此架構與主從式架構最大的差異。Napster 爲第一代點對點傳輸架的代表。

歸納此一架構的優點如下：

- 不需處理檔案傳輸，伺服器負載減輕許多。
- 一般狀況下，因爲來源多所以傳輸速度提升許多。
- 通訊協定簡單，用戶端不需維護其他用戶端資訊。

歸納此一架構的缺點如下：

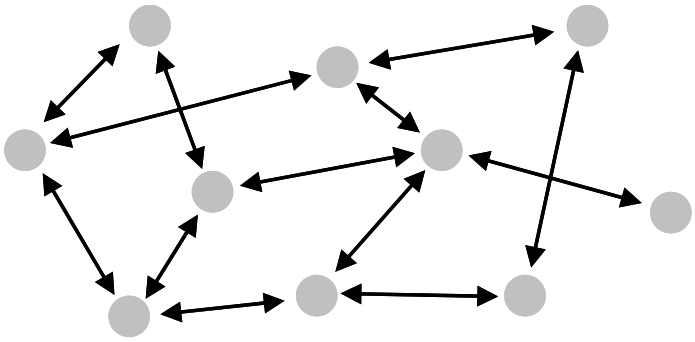
- 已經證實爲非法。法院判 Napster 有大量的侵犯版權行爲發生，下令其改進[30]。
- 因爲有固定的索引伺服器，容易成爲攻擊目標，一旦索引伺服器被攻破或者故障，則搜尋功能完全喪失。



圖表 6 第一代點對點傳輸架構

2.2.3 第二代點對點傳輸架構

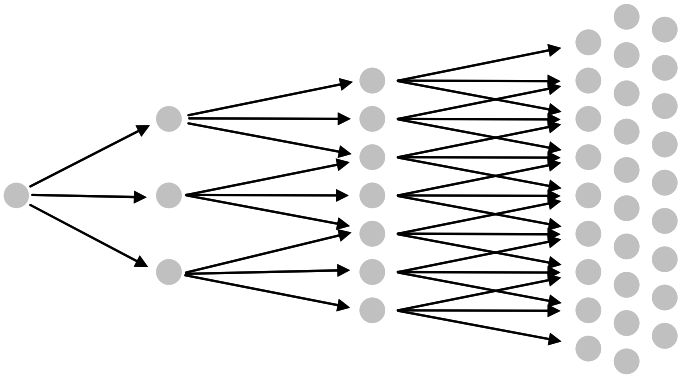
第二代主要是爲了解決第一代中索引伺服器所引發的各種問題，尤其是違法問題。此架構下只有一種元件：點。所以架構中的服務必須由各點互相合作，因此出現鄰居(Neighbor)的概念，因爲網路資源分散各處，所以此架構又被稱爲分散式服務。每點除了爲接收端之外，亦會建立本地端可分享資源的索引表，提供其他點搜尋。加入此類架構，必須藉由已在架構下的點輔助資訊交換認識其他鄰居。當點搜尋資料時，只需將搜尋封包傳送給所認識的鄰居。



圖表 7 第二代點對點網路傳輸架構

分散式服務起初是以 Gnutella 爲龍頭，其搜尋時將搜尋封包傳送給所有鄰居，然後鄰居各自回應原始點，並轉送搜尋封包給第二層鄰居，如此遞迴下去。此種作法，造成相當嚴重的搜尋洪流(Query Flooding)。後來 FastTrack (即 KaZaA 的底層技術)迅速掘起取代其地位，改爲搜尋封包在點間傳送，直到找到檔案實體才將搜尋結果回傳給原始點，並非每一點都做回應。而這種技術仍相當不便，容易產生無用的搜尋迴圈，當大量

搜尋同時執行將造成網路效能大幅下降。此問題最簡單的處理方式就是限定搜尋深度(Gnutella 限制往下搜尋七層)，但是這個作法讓搜尋只限定於邏輯網路的某些區域[29]。



圖表 8 Gnutella 搜尋邏輯

但是限制搜尋深度仍然消耗很大的網路成本。以 Gnutella 來說，T 爲搜尋深度，N 爲每個用戶端相連的個數，則每次搜尋所影響的用戶端個數如圖表 9[25]。

	T=1	*T=2*	*T=3*	*T=4*	*T=5*	*T=6*	*T=7*
N=2	2	4	6	8	10	12	14
N=3	3	9	21	45	93	189	381
N=4	4	16	52	160	484	1,456	4,372
N=5	5	25	105	425	1,705	6,825	**27,305**
N=6	6	36	186	936	4,686	**23,436**	**117,186**
N=7	7	49	301	1,813	**10,885**	**65,317**	**391,909**
N=8	8	64	456	3,200	**22,408**	**156,864**	**1,098,056**

圖表 9 Gnutella 的搜尋影響

歸納此一架構的優點如下：

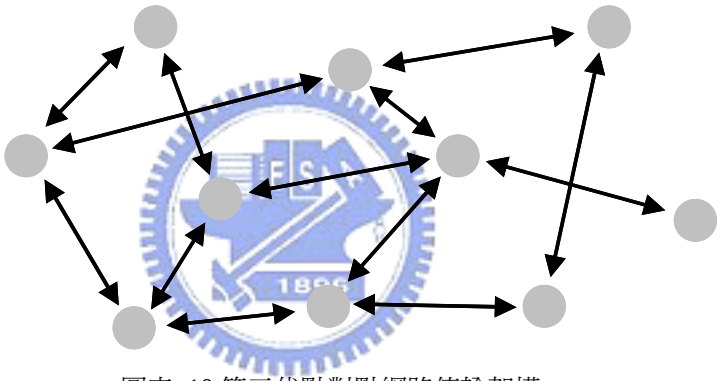
- 沒有索引伺服器的限制。
- 目前美國法院判定法律上合法[29]。

歸納此一架構的缺點如下：

- 雖然利用限定搜尋深度，但是依然造成網路壅塞。而搜尋封包在網路中來回傳送，造成搜尋時間過長
- 只能做到區域性搜尋(Local Search)。
- 較多人知道的點往往成爲熱點(Hot Point)。造成該點負載遠高於其他點。
- 如果預知點剛好都不在網路架構下，則等於無法加入該網路架構。

2.2.4 第三代點對點傳輸架構

第三代是近期的主力，主要是因爲加入分散式雜湊表(DHT，Distribute Hash Table)的概念，不再是第二代中雜亂無章的封包轉送，取而代之的是有系統、漸進式的尋找。此架構下亦只有一種元件：點。不過每個點將會擁有一個唯一的用戶端編號。而所有的資源也都有自己的資料編號，如第一章中所述，所有資料編號資訊將會應對到有特定用戶端編號的點上。當點 A 加入架構時，點 A 必須知道至少一個以上已在架構下的點 B，藉由 B 交換資訊，A 將會找到一些點集合(Peer set) {S}。而{S}中所有點的用戶端編號和 A 的用戶端編號最爲接近，亦可稱爲 A 的鄰居。除了找到鄰居之外，也會找到 A 必須負責的所有資料編號。



圖表 10 第三代點對點網路傳輸架構

搜尋時，點會計算出所需資訊應有的資料編號，然後藉由該資料編號計算出屬於哪一個用戶端編號，然後從所有鄰居中找出最接近該用戶端編號的一個送出查詢封包。基本上，此法解決了搜尋洪流，同時也不需限制搜尋長度，相對地，可做到廣域搜尋(Global Search)。再者良好的鄰居選擇策略可以降低搜尋跳躍數目，類似架構下如CAN需要 $O(N^{(1/d)})$(N：點總數、d：維度)、Chord、Pastry、Kademlia等都可以降低到O(LogN) 。較詳細技術概念將在下一章解說。

第三代點對點傳輸架構則是以 EDonkey- Kademlia 及 Morpheus 爲代表，此外還有一些較小的獨立軟體開發商，配合特殊的改善方式，使這些工具比以前更有效率。根據網路監視公司 BayTSP 的使用者數量評比，EDonkey 超越 KaZaA 成爲全世界最受歡迎的點對點傳輸軟體[29]。

歸納此一架構的優點如下：

- 解決搜尋洪流。
- 可做到廣域搜尋。

歸納此一架構的缺點如下：

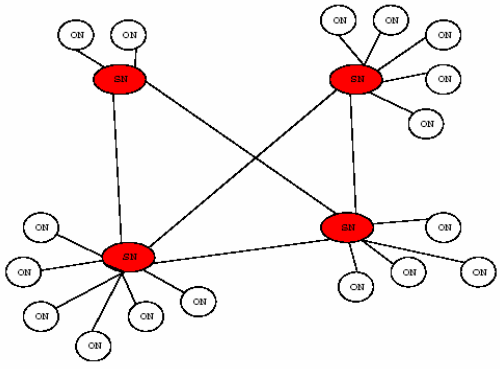
- 每點的責任相對變大，點離開將會造成搜尋失敗率變高。而且必須花額外的頻寬來維護點與資料的關係。
- 部分架構加入其他技巧如樹狀搜尋、快取等方法加速搜尋，但同時也增加許多維護訊息。

2.2.5 其他點對點網路傳輸架構

除了前述的架構之外，還有一些其他的架構，其中最有名的就是 KaZaA、Bittorrent。這兩種架構的效能相當受肯定。下面將說明其概念。

KaZaA

KaZaA 到目前爲止尚未公開其架構，現有的文獻是經由分析該軟體的行爲，反推而得的結果。但是推測的架構的確是可行的，而且由非官方單位依其架構開發的軟體 KaZaA-Lite 現在也廣爲流行。此架構下也是只有點，但是點會因傳輸能力而被區分爲 ON(Ordinary Node)、SN(Super Node)拓樸如圖表 11，ON 爲一般用戶端，SN 爲網路資源大的用戶端(如頻寬大)。不過使用者無法直接得知本身爲 ON 還是 SN，因爲 ON 可能因爲被整個系統判定爲高資源者，進而轉變成 SN 來處理較多的事件[15][23]。



圖表 11 KaZaA 網路架構

事實證明這樣的機制，的確可使傳輸增快，以此架構產生的 KaZaA、Skype、KaZaA-Lite 等，都成了目前熱門的點對點軟體。新用戶端 A 加入整個網路架構時，其本身先爲一個 ON，且 A 需要知道至少一個執行中 SN 的網路資訊，進而得到歸屬 SN。當連結成功後，A 會將自己要分享檔案的媒體資料(如檔名、長度、雜湊值等其他資料)上傳到所屬的 SN。搜尋時，ON 只將搜尋封包傳送到 SN，SN 除了搜尋本身所有已知的媒體資料，也可能對其他 SN 做搜尋轉送，最終由 SN 再把搜尋結果送回 ON[7]。

歸納此一架構的優點如下：

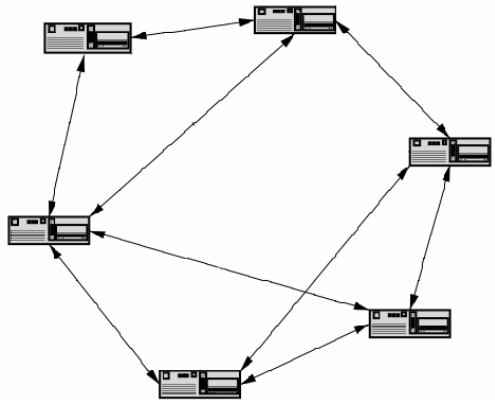
- 較能充分利用網路資源，搜尋速度快。
- 額外維護點之間關係的頻寬較低。

歸納此一架構的缺點如下：

- SN 服務品質直接影響所屬的 ON。
- 被選爲 SN 的用戶端，負載會增加許多。

Bittorent

Bittorrent 沒有提供搜尋功能，使用者需自行尋找所需種子(SEED，.torrent 檔)，種子中包含追蹤者(Tracker)網路位置、最初來源網路位置、檔案雜湊值等檔案相關資訊，利用種子才能下載所需檔案。Bittorrent 的架構由追蹤者、種子來源、點組成。追蹤者在 Bittorrent 佔最重要的角色，追蹤者會輔助檔案尋找，因爲追蹤者紀錄哪些點有所需資料的片段可供下載。種子來源提供種子取得，可以是網站論壇或使用者之間互相交換。點則是每個用戶端，不僅爲接收端，同時也爲來源端。



圖表 12 BT 傳輸架構

Bittorrent 的加入，不需要透夠複雜的溝通，只要得到種子即可。然而搜尋即爲種子尋找。取得種子後，Bittorrent 軟體會解析種子的內容，取得追蹤者的網路位置，並與追蹤者溝通，然後詢問追蹤者是否有擁有資源的來源點，獲得來源點的網路位置後，點就開始向這些點要求傳輸。

網路監視公司 BayTSP，以流量監測的方式觀察網路，若以總流量來評比，Bittorrent 是流量最大的點對點應用軟體[29]。

歸納此一架構的優點如下：

- 不需要繁雜的系統維護。
- 實際網路中流量最大。

歸納此一架構的缺點如下：

- 缺少搜尋的機制，必須由使用者自行尋找種子。
- 檔案資源的分享者，必須自行尋求穩定的追蹤者。

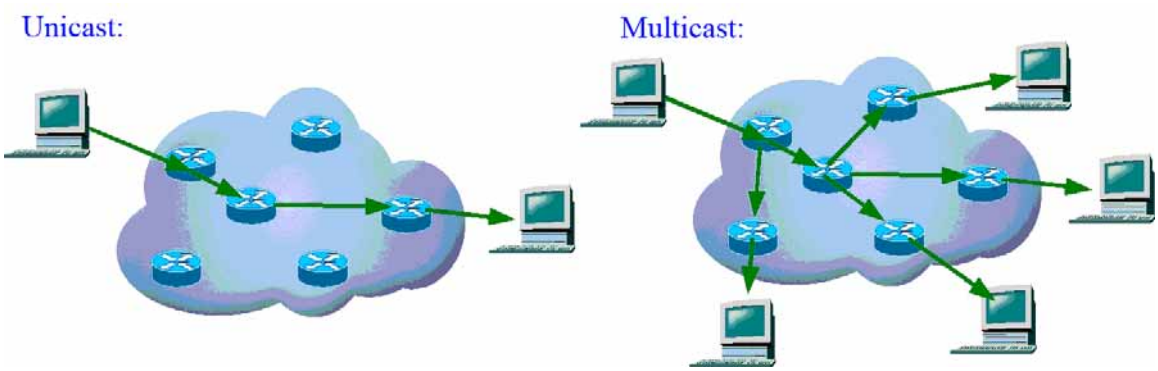
2.3 多址傳輸（Multicast）與路由（Routing）

多址傳輸(Multicast)，泛指 IP Multicast，這種機制可以提昇傳輸速率並減少網路壅塞的現象。若是使用傳統 IP 傳輸，多個用戶端向同一用戶端要求同一份資料，則該份資料必須反覆的由來源用戶端傳送給各個用戶端，來源端的傳輸速率與負載都是瓶頸。由於點對點應用的日漸流行，也有研究利用應用層軟體來做到多址傳輸。另外，路由(Routing)在傳統網路下，都是由路由器來選擇路由路徑，然而在點對點的架構下亦可利用應用層軟體來輔助封包路由。下面小節我們將概述 IP Multicast、應用層多址傳輸(Application Layer Multicast)、IP Routing 以及應用層路由(Application Layer Routing)。

2.3.1 IP Multicast

IP Multicast 的精神就是將一個來源封包，同時送給一個用戶群組(host group)，而用戶群組的大小，主要是依照硬體規格與網路架設而定。IP Multicast 希望做到流量最佳效能(best-efforts)。對用戶端來說，用戶可以隨時加入或離開某一用戶群組，使用上和傳統

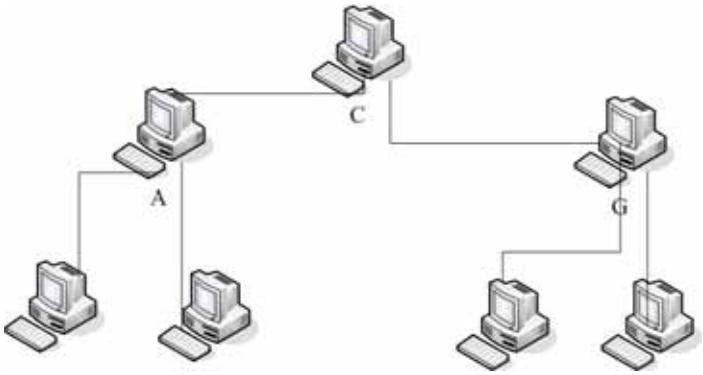
網路沒有不同。IP Multicast 需要硬體配合，目前一般的標準路由器皆有此項功能。佈設 IP Multicast 於一網域中，至少要有一台提供 IP Multicast 的路由器，而用戶群組必須向其註冊，用戶群組的成員不一定是用戶端，可爲閘道器(Gateway)或者是另一路由器。提供 IP Multicast 的路由器越靠近分枝匯流處，越能減少頻寬浪費，總體效能就越顯著。IP Multicast 通常多用於串流(Streaming)等同步性質的應用，如網路節目、網路廣播等。



圖表 13 Unicast 與 Multicast

2.3.2 應用層多址傳輸（Application Layer Multicast）

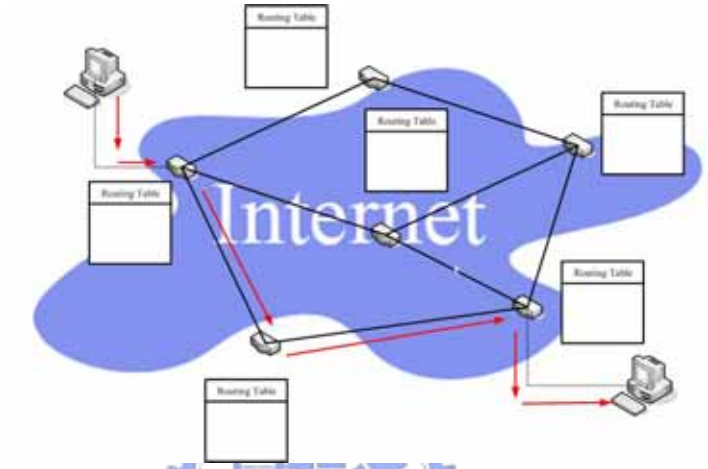
有鑑於現在的用戶端設備日益增強，目前已有實際軟體以用戶端來做影音的應用層多址傳輸如 coolstreaming[24]，這種應用就是以點對點架構來實現。觀念和 IP Multicast 類似，只是把負責多址傳輸的路由器改爲用戶端設備。最大差別在於來源選擇，系統必須提供一套機制來輔助選擇某一用戶爲新進用戶端來源。此外這類軟體也會充分利用每個用戶端，除了爲接收端外，也爲其他用戶端的來源端，藉此做到負載平衡。



圖表 14 Application Layer Multicast

2.3.3 IP Routing

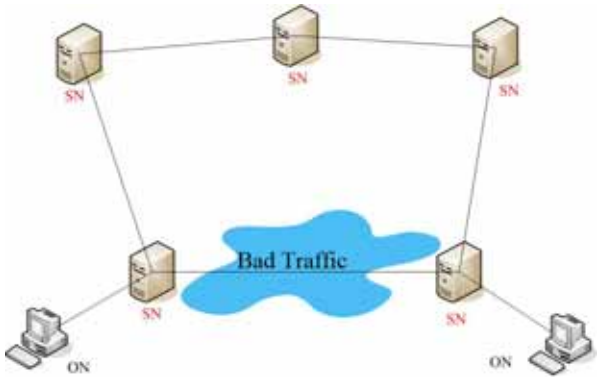
路由(IP Routing)，即封包傳送的路徑選擇，一般都是交由路由器處理，路由器之間透過各種不同的演算法如 RIP、IGRP、OSPF、BGP 等，計算出路由表，而傳送封包時就依照路由表中的最佳路徑來傳送。一般來說，路由表中所指示的路徑，都是擁有某種特性的最短路徑，如跳躍數最低、流量最低等。



圖表 15 IP Routing

2.3.4 應用層路由（Application Layer Routing）

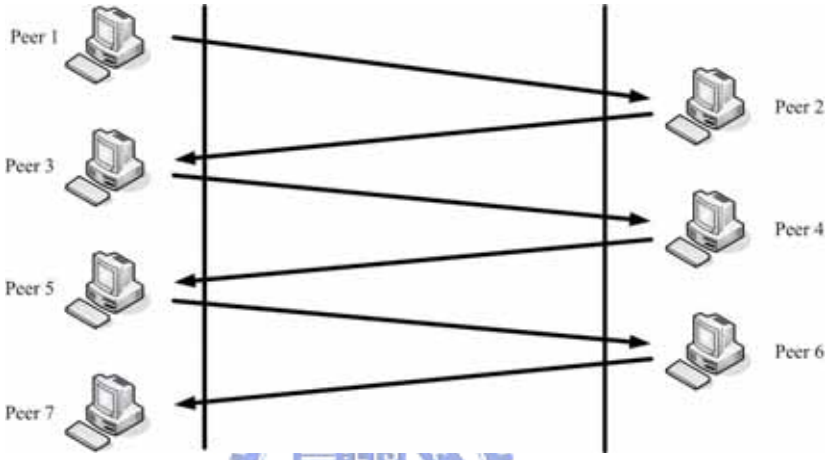
應用層路由(Application Layer Routing)顧名思義就是利用應用層的程式，來輔助傳送封包，最有名的就是即時通訊軟體 Skype。如圖表 16，不同 SN 下的 ON 溝通時，SN 會因爲最短路徑的網路品質不佳，而選擇走較遠的路徑，很多實驗也證明這種方法的傳送品質優於其他產品[17]。



圖表 16 Skype 封包路由

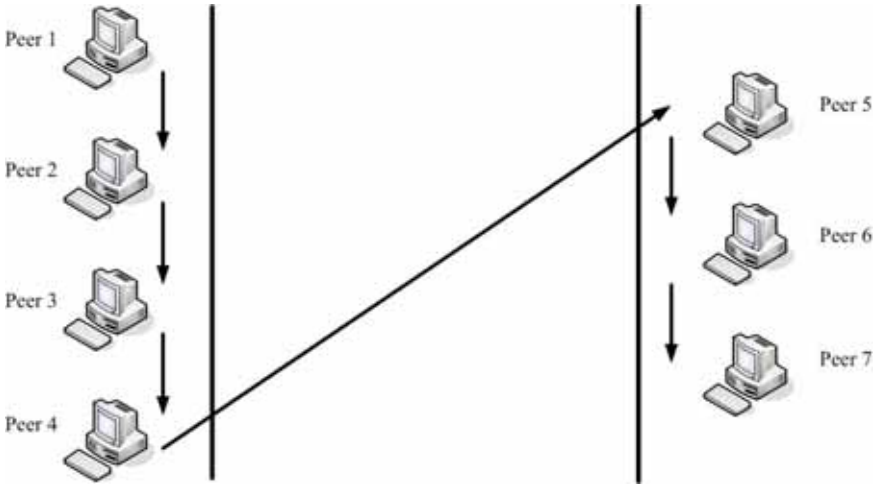
2.4 地域(Locality)

地域的功用，是爲了使得邏輯網路接近實體網路[1]。因爲傳遞延遲還是受到實體網路的影響，所以如果第三代佈設時，點的用戶端編號隨便給定，很有可能造成用戶端編號相近的兩個點實際上是相距很遠的。檔案搜尋時，邏輯上的鄰居互相傳遞訊息，最差的情形如圖表 17，假設搜尋次序是點 1、點 2、點 3、…、點 7。



圖表 17 無 Land Mark 的最差情形

使用地域技術後，理想上得如圖表 18 的效果，大幅減少長距離傳輸，以加快速度。目前常見的地域技術有兩種，一種是利用網路位置定位(IP Locality)，另一種則是 RTT 技術。下面兩個小節將一一介紹。



圖表 18 具 Land Mark 的理想情形

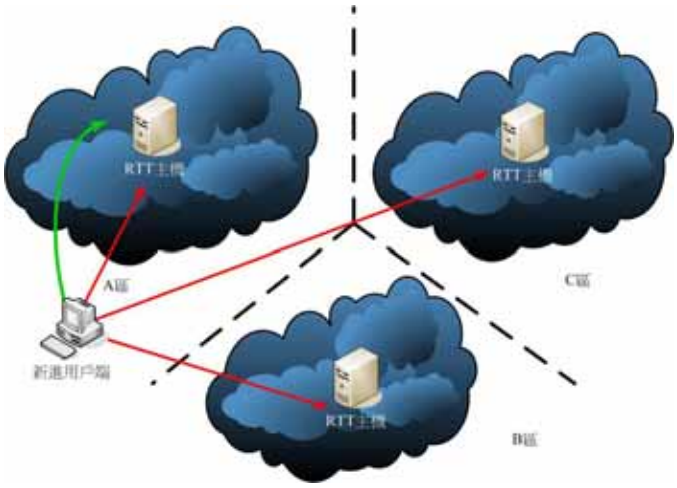
2.4.1 網路位置定位(IP Locality)

網路位置定位是利用網路分級的特性，譬如國立交通大學、國立中正大學都屬於 class B 的網路。簡單的說，國立交通大學網域中，所有的網路位置皆爲 140.113.X.X；同理國立中正大學網域中，所有的網路位置皆爲 140.123.X.X。其背後的假設爲在正常狀況下，國立交通大學中所有網路設備，實體上是相接近的，互相溝通時所需經過的硬體設備也比較少，自然延遲就比較小。所以網路位置定位就是利用這種假設，任何兩台 140.113.X.X 的網路設備所得的用戶端編號差值，盡可能小於某一 140.113.X.X 和某一 140.123.X.X 的用戶端編號差值，此種方法不需要額外的設備即可實施。但是也相當容易造成誤差。因爲事實上，有鑑於 IP 分發已經接近飽和，所以申請單位不可能再拿到 class A、class B 級大區域網路，甚至可能分配到幾個 IP 而已。舉例來說，如果 163.101.7.X 分配給高雄的某單位、而 163.101.8.X 分配給台北的某單位、163.101.9.X 分配給台中的某單位，以網路位置定位的觀點來看 163.101.7.X 和 163.101.8.X 比較接近，但是實際上 163.101.7.X 和 163.101.9.X 卻是比較接近，在這種情況下，網路位置定位的誤判就會很嚴重。

2.4.2 RTT 技術 － 地標(Landmark)

RTT 全名爲 Round Trip Time，就是以 ICMP 中 Ping(Packet Internet Groper)的回應時間來輔助定位，此種作法就是在網路中，平均分散多台主機以輔助分區，其行爲有如地標，所以此種技術亦稱爲地標(Landmark)技術。當一個新的用戶要加入邏輯網路時，會先對各地標主機做 Ping 的動作，然後利用回傳結果判別應屬於哪一個分區，然後由邏輯網路或者自行算出一個唯一用戶端編號。而此用戶端編號值會和同分區的其他用戶端編號值相近；相反的，不同區域的用戶端編號值就會相差較大。如圖表 19 所示，新進用戶端在加入時，會先向三個地標主機做 Ping 的動作(紅色線直線)，得到結果後，發現與 A 區較爲接近，所以加入時其用戶端編號會盡量接近 A 區的用戶端編號。

此一方法，通常可以取得較精確鄰近關係。但是相對的，RTT 技術也需要有額外的網路設備，另外，當時的網路流量、地標的分佈也會影響 RTT 技術的結果。已有論文嘗試加入 RTT 技術，所得的結果相當顯著[1]。

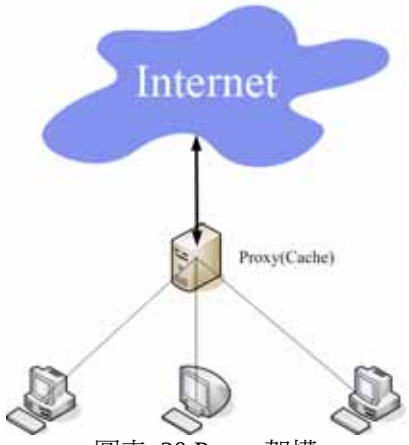


圖表 19 RTT 技術

2.4.3 代理(Proxy)

代理可做到許多功能，如多位使用者透過單一連線存取網際網路的存取控制(Access Control)、防止外部網際網路未經授權進入區域網路內(Firewall)等。代理和本論文最有關的功能是藉著將網頁內容及經常存取的資訊存入快取(Cache)，降低本地端網路與網際網路間的資料傳輸量。

在 Kademlia 中亦有類似的觀念。當查詢第一次經過用戶端時，該用戶端除了轉送查詢之外，亦會記住該次查詢的結果，當相同的查詢經過該用戶端，則直接回覆查詢。利用此法，查詢流量可大幅降低[5]。其實亦可實作檔案實體代理，但是這樣代理用戶端將有違法的可能。



圖表 20 Proxy 架構

不過代理並非無缺點，因爲網路的不穩定性，造成代理資料有效性隨著時間改變，所以代理用戶端必須有應對策略，可以主動重複查詢，亦可利用逾時(Time out)取消。通常後者是容易設計且符合經濟效益的，因爲熱門查詢也是有時間性的，不會一直持續。

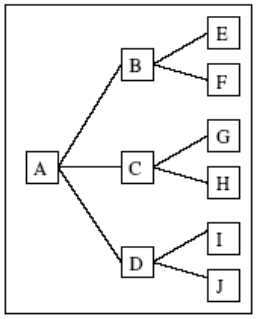
第三章 相關研究

本章節將介紹與本論文相似的一些研究。3.1 RI 是介紹查詢轉送路徑選擇概念。3.2 CAN 是最初導入第三代的定位觀念。3.3 Chord 是本論文修改的主軸論文。3.4 Pastry 帶入前置共通(prefix common)的搜尋觀念。3.5 Kademlia 是介紹目前最多人使用的點對點傳輸架構-Emule (Edonkey)的第二代網路架構。

3.1 Routing Index

點對點傳輸進入第二代以後，鄰居的觀念出現了。簡單的說，認識鄰居越多，搜尋結果也越多，相對的，維護鄰居的負載也跟著升高。另外，如果每次搜尋資料都對全部鄰居做轉送查詢的動作，不僅造成網路壅塞，也可能造成搜尋迴圈等問題。

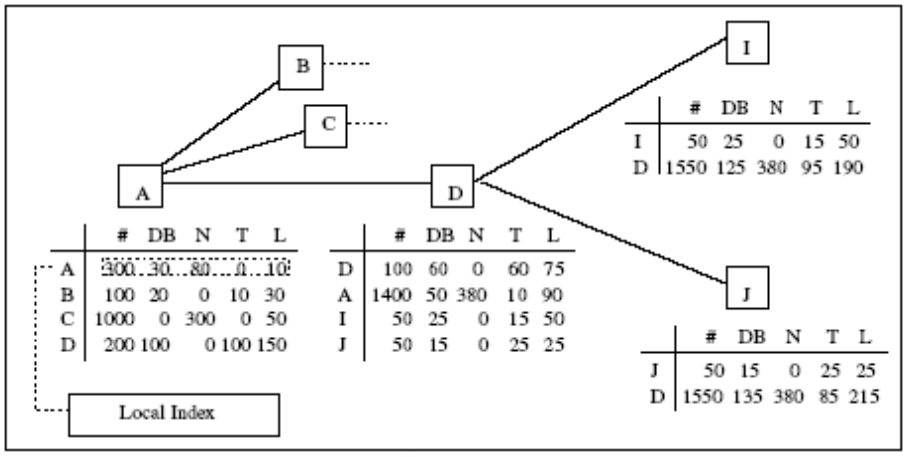
Path	# docs	Documents with topics: DB	N	T	L
B	100	20	0	10	30
C	1000	0	300	0	50
D	200	100	0	100	150



圖表 21 RI 架構

RI 的觀念則是選出最適合的鄰居來做轉送搜尋。何謂適合不適合呢？RI 的概念就是對資料分群並做數值統計，利用統計的數值來做合適的標準。RI 的拓樸是一種樹狀架構如圖表 21，每個用戶端都是樹上的節點，利用樹的特性即可解決搜尋迴圈的問題。每個用戶端只要維護邏輯上相連的節點即可。以實作來說，每個用戶端必須維護一張轉送表，其中紀錄本身以及轉送可得資料的統計數值。搜尋某一分類資料時，盡量以統計數值大者來轉送。

以圖表 22 說明，分類爲 DB(database)、N(network)、T(theory)和 L(language)，單位爲文件數目。用戶端 D 需要維護的轉送表，第一列即爲本身擁有的資料：DB 有 60、N 有 0、T 有 60、L 有 75，第三、第四列分別爲可從用戶端 I、用戶端 J 取得的文件。而第二列爲可從用戶端 A 取得的文件，文件數量相對大上許多。仔細對照用戶端 A，會發現，其實用戶端 D 中可從用戶端 A 取得的文件，除了用戶端 A 本身之外還加上用戶端 B 與用戶端 C 的文件。用戶端 D 搜尋時，會依照 Goodness 來選擇轉送的用戶端，Goodness 其實就是一條公式，公式是利用表中的值來做運算，而最簡單的就是取量化值最大的用戶端來轉送。如果是在這情形下，用戶端 D 搜尋 T 時，轉送時會選擇 J，反而不是總文件量最大的 A。

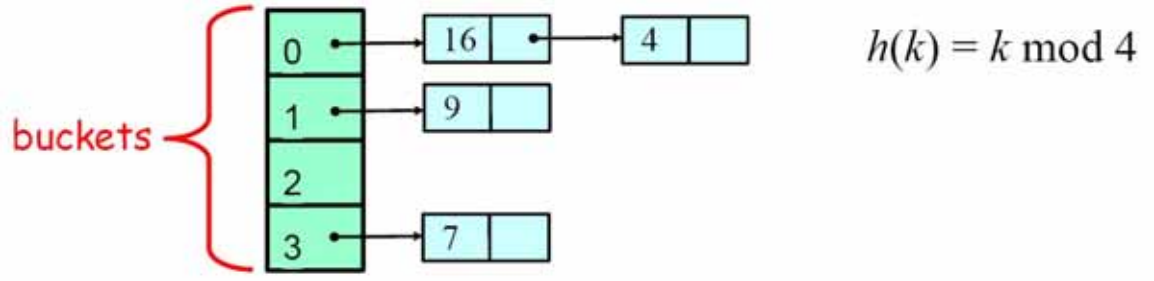


圖表 22 RI 各用戶端資料表的關係

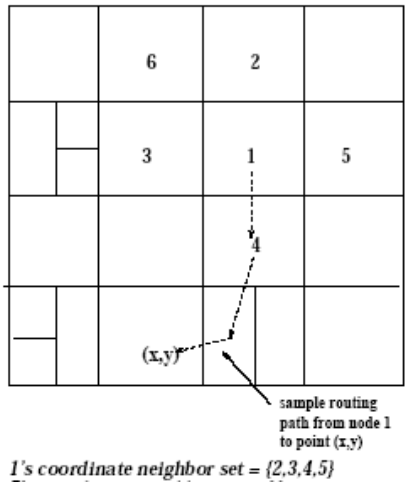
3.2 CAN

搜尋一直是點對點傳輸的最大問題，尤其是搜尋不到時，搜尋訊息交換將會到達最大浪費。這延伸出一個問題，什麼時候確定沒有資料了，不需再搜尋下去。第三代點對點傳輸的最初設計就是爲了解決這個問題，而最初的想法就是 CAN。CAN 的想法很簡單，就是把搜尋問題轉變成定位問題，也就是把搜尋的想法改成雜湊(Hash)的想法[1]。雜湊主要由兩個部分組成：一個良好的雜湊函式(Hash Function)以及一連串規劃好的邏輯位置空間(Buckets)。資料置入儲存空間時，並非任意置放，而是置放前利用資料特徵(key)透過雜湊函式算出對應的邏輯位置，然後將資料置放在計算出的邏輯位置空間。所以搜尋時，只要算出邏輯位置，即可快速取得資訊。同一個邏輯位置可能有多個物件，

在同一個邏輯位置的尋找可以是線性尋找或者其他演算法，但是基本上，同一個邏輯位置中的數量少，搜尋自然快。



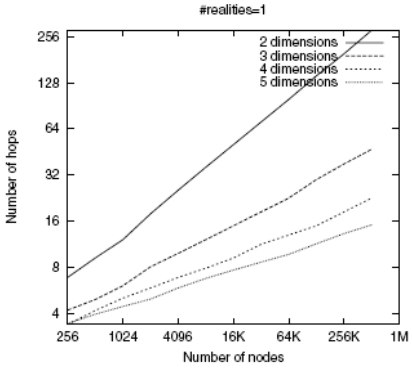
轉換的觀念很簡單：把點對點網路架構下的所有用戶端當成一連串規劃好的邏輯位置空間，邏輯空間的大小 M 取自於系統設定。所以每個用戶端都需擁有唯一的用戶端編號 ID(0 <= ID < M)。檔案 F 要分享給整個網路架構時，必須先算出 F 的資料編號，然後將 F 取 M 的餘數 R。並將 F 的相關資訊放置在 ID 等於 R 的用戶端上。所以，任一用戶端搜尋檔案 F 時，只要詢問 ID 等於 R 的用戶端是否擁有該檔案即可。



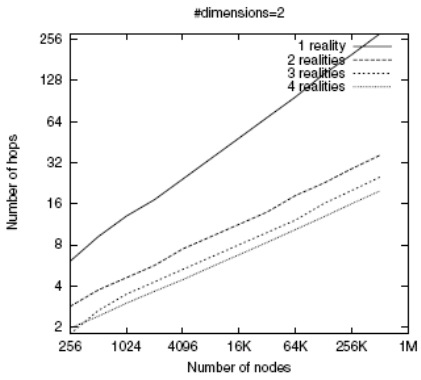
圖表 23 CAN 的搜尋邏輯

舉二維CAN來說明。如圖表 23，基本上整個邏輯網路，如同棋盤一樣，切割成一個一個單位方形，所以也有人稱爲棋盤式網路拓樸，每一個用戶端a 都屬於某一個座標(X_a , Y_a)，而每一個座標就如同前述的邏輯位置空間。相同地，每一個分享檔案f ，都會經由一套雜湊公式計算出所屬的邏輯位置空間(X_f , Y_f)。所以f 的檔案資訊將會由座標位置爲(X_f , Y_f)的用戶端維護。

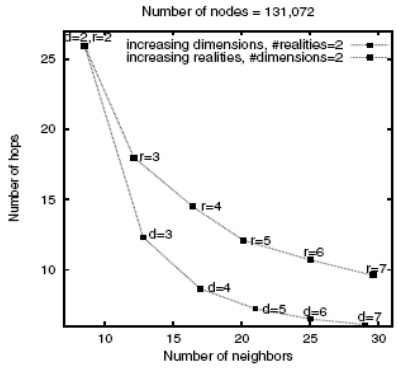
當一個新用戶端a 加入CAN時，將會藉由一個已在CAN網路中的用戶端c 輔助，所以c 發出加入訊息由自己座標(X_c , Y_c)往(X_a , Y_a)移動，一次一個單位要移動abs(X_c - X_a) + abs(Y_c - Y_a) (abs絕對值)次，通知(X_a , Y_a)上下左右四個鄰居，各自更新相對鄰居。搜尋時也是一樣，算出目標座標後，一次一單位的移動，最終到達目標座標。下面圖表 24、圖表 25 分別表示CAN的效能(dimension表示邏輯網路的維度，reality表示邏輯網路的個數)。基本上維度越高，實體個數越多，效能就越高。但是維度超過四之後，效能的進步有限；而實體個數超過三之後，進步也有限。圖表 26 則是固定用戶端個數分別對維度與實體做改變而得的結果[1]。



圖表 24 CAN 維度與路徑長度關係



圖表 25 CAN 網路實體個數與路徑長度關係



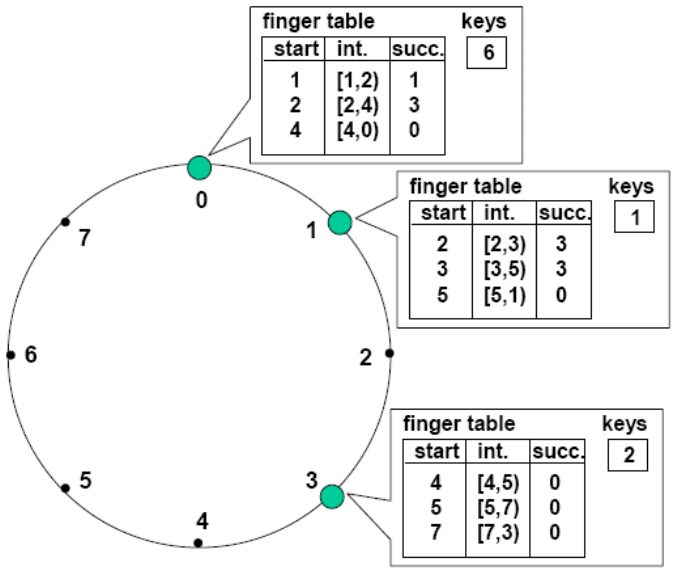
圖表 26 CAN 的實體個數、維度與路徑長度關係

CAN所需維護的鄰居個數相當少，就是 2d個(d :維度)，所以維護鄰居的網路成本很低。但是因爲只維護相鄰用戶端，所以查詢轉送時一次只跳過一個用戶端。用戶端之間的查詢訊息，很直覺地需要O($N^{(1/d)}$)(N：爲用戶端總數，d爲維度)，基本上還是相當浪費網路成本。所以後來的論文，多是在CAN上做修改以增進效能，最有名的就是Chord。

3.3 Chord

CAN的論文中，表示維度大到一個限度之後，就無法再降低搜尋路徑長度[1]。理由很簡單，因爲CAN每一個維度至少都要做一次以上的移動，所以當維度爲d時，至少要移動d次(除非該維度座標值恰好等於目標座標在該維度的值，但是機率相當低)。CAN有這個先天的障礙，所以無法再將路徑長度縮小。Chord爲了解決這個問題，反而以一維的環狀(Ring) CAN來做拓樸設計，但是邏輯搜尋上卻是以二元搜尋(Binary Search)的概念來降低搜尋路徑。所以Chord可以將搜尋的最大跳躍次數限制在$\log_2 N$ (N爲用戶端總數) [3]。

要做到二元搜尋，環狀拓樸上每個用戶端必須維護和本身相關的資訊，除了本身的用戶端編號外，還得記住一張對應表(Finger Table)紀錄和自己編號相距(N / 2)、(N / 4)、(N / 8)、…、(N / 2^k)的用戶端編號，共$\log_2 N$個。環狀拓樸上不可能一開始就到達用戶端總數，所以會有跳號的情形，基本上，遇到跳號就往後找到一個存在的用戶端，這個方法稱爲找繼承人(successor)。以圖表 27 [3]說明：



圖表 27 Chord Finger Table

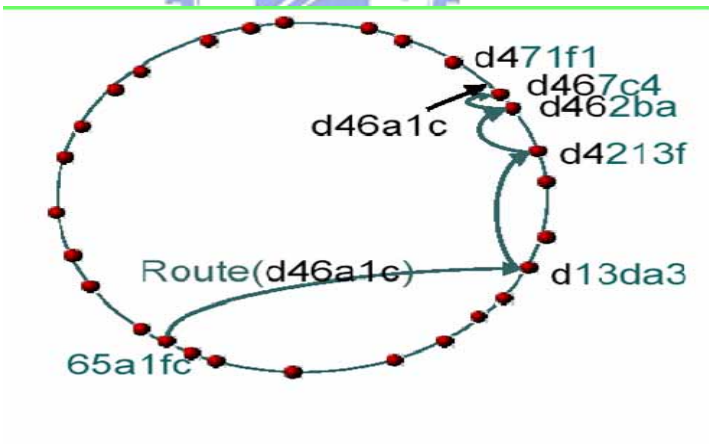
例子中的環狀拓樸，最大用戶端數爲 8，即N = 8。每個用戶端維護一張對應表與和對應的關鍵值(Keys)表。關鍵值表就是記錄所有雜湊值和該用戶端編號相等檔案的檔案資訊。對應表就是記錄前述的$\log_2 N = k = 3$ 個用戶端資訊，所以對應表中有 3 列，每欄

代表意義：i表示第幾列，0 <= i < 3，int.表示區間，即表示編號相距 2^i 至 $2^{(i+1)}$，start區間的起始用戶端編號，而succ.表示這個區間中實際start的用戶端編號。succ.可能不屬於int.，因爲int.中沒有實際存在的用戶端。

一個新用戶加入Chord，必須透過一個已在Chord網路的用戶端輔助，找到邏輯上相關的鄰居，並觸發更新。Chord論文中指出需要約$(\log_2 N)^2$則更新訊息。而搜尋需要$\log_2 N$個搜尋轉送[3]。

3.4 Pastry

Pastry是另一種CAN的變形，同樣也是爲了降低CAN的路徑長度。Pastry也是採用一維的環狀拓樸，但是其搜尋的想法，則是採用前置共通(prefix common)的概念。以圖表 28 做簡單的說明，用戶端編號長度固定皆爲l，而搜尋目的的編號爲$(n_1,n_2,n_3,\ldots,n_l)$。搜尋時，從座標$(m_1,m_2,m_3,\ldots,m_l)$開始，第一次跳到$(n_1,m_2,m_3,\ldots,m_l)$，第二次跳到$(n_1,n_2,m_3,\ldots,m_l)$ 路徑每增加一，就多一位前置碼相同，所以路徑長度最長爲l[4]。



圖表 28 Pastry 搜尋

前置共通，Pastry的作法則是給定一個預先設定的單位值b ，建議給定 b 爲 2 或 4 。舉例說明，假設Pastry邏輯網路的參數b = 2、l = 8，此時每個用戶端都要維護一張表如圖表 29，而該表又可分爲三張子表：Leaf set、Routing table、Neighborhood set。Leaf set是記錄編號和本身編號最靠近的一些用戶端資訊，其中又分較大和較小兩部分，各l / 2 = 4 個。Neighborhood set則是紀錄實體接近的一些用戶端資訊，共l = 8 個。Routing table記錄轉送的用戶端$\log_2{}^b N$列 2^b欄，而該子表的設計是有特殊的技巧，藉以加速查詢

轉送，每一個表中元素都可用一個公式M-C-R表示其對應的編號(M：前置相符的編號，C：欄位號碼，R：剩下部分)。以Routing table中第二列第一欄來說明 10-1-32102，M爲10和本身編號10233102的前兩個前置共通碼相同，C爲1表示第一欄，R爲剩下的32102。

NodeId 10233102

Leaf set	SMALLER	LARGER	
10233033	10233021	10233120	10233122
10233001	10233000	10233230	10233232

Routing table

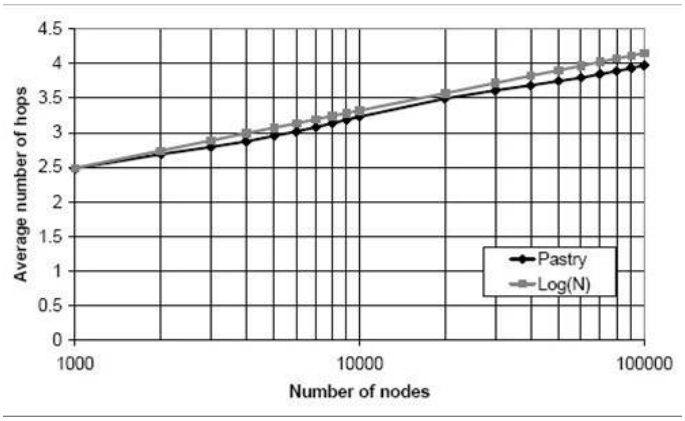
-0-2212102	1	-2-2301203	-3-1203203
0	1-1-301233	1-2-230203	1-3-021022
10-0-31203	10-1-32102	2	10-3-23302
102-0-0230	102-1-1302	102-2-2302	3
1023-0-322	1023-1-000	1023-2-121	3
10233-0-01	1	10233-2-32	
0		102331-2-0	
		2	

Neighborhood set

13021022	10200230	11301233	31301233
02212102	22301203	31203203	33213321

圖表 29 Pastry 各用戶端要維護的表

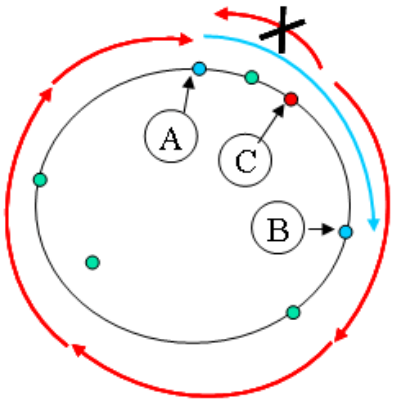
一個新用戶加入 Pastry，必須透過一個已在 Pastry 網路的用戶端輔助，找到邏輯上相關的鄰居，並觸發更新。而搜尋時，主要分爲三個階段，第一階段先尋找 Leaf set 內是否存在？存在即回覆，否則進入第二階段。第二階段依照 M-C-R 的方式找到對應的元素，元素存在即回覆。否則進入第三階段。第三階段則是從三張子表中找出數值上最接近目標編號的用戶端來轉送。雖然多了許多的步驟，可是效能上可以表現得更好，該論文的提出測試數據如圖表 30，可以做到小於 logN (N：爲同時在該邏輯網路下的用戶端總數) [4]。但是相對地，Pastry 要維護的相關用戶端多上許多。



圖表 30 Pastry 的效能測試

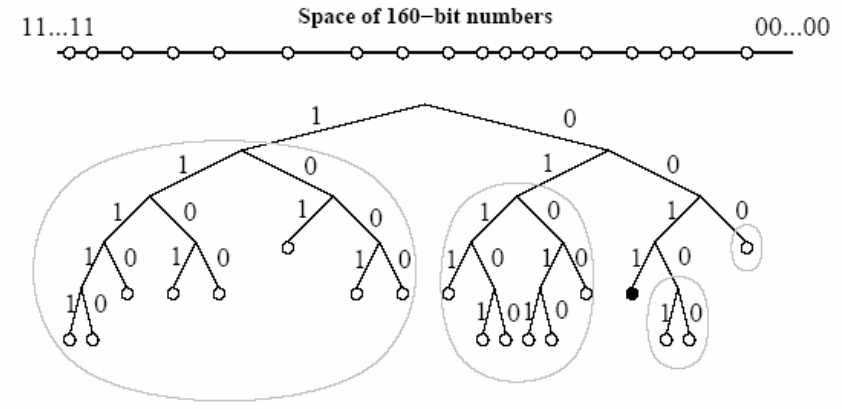
3.5 Kademlia

Kademlia 亦是另一種 CAN 的變形，採用變形二元搜尋樹(Binary Search Tree)的概念，使得搜尋跳躍數可以更低。Kademlia 指出 Chord 架構有一個缺點，即只能單向搜尋。以圖表 31 的例子，邏輯上 C 和 A 較近，可是 C 要到達 A 卻需要走比 B 還要遠。Kademlia 提出對稱的想法，使得邏輯的搜尋可以雙向，如此一來，邏輯上較近的，搜尋的路徑也會較短[5]。



圖表 31 Kademlia 提出 Chord 的缺點

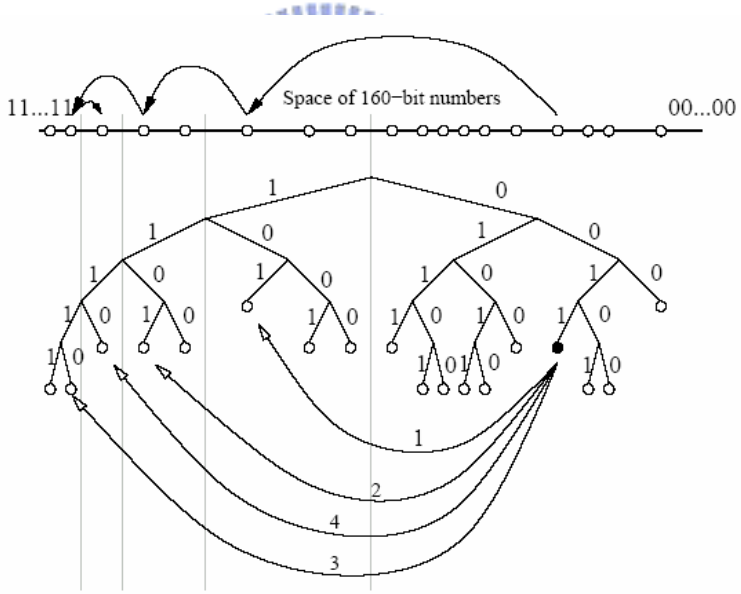
Kademlia概念上建立一棵二元樹(Binary Tree)，每點都是樹的葉(Leaf)，而在正常二元樹中搜尋，只要$\log_2 N$。但是爲了避免這棵樹太過龐大，則採用前置共通的想法來縮減。而每點維護的鄰居，並非只有母節點(Parent Node)和子節點(Child Node)而已，還要記住以自己爲基準的相對子樹中部分節點，每個子樹記錄k個點，藉此可做邏輯上的雙向跳躍。圖表 32 表示黑點需記錄的相對子樹。



圖表 32 相對子樹

一個新用戶加入Kademlia，必須透過一個已在Kademlia網路的用戶端輔助，並觸發更新。更新時，同時也要維護邏輯二元樹的完整性。搜尋時，同樣是利用遞迴的作法，每個回合都判斷目標節點的所在子樹，然後做搜尋跳躍，直到找到為止，需要O(log(N))，為了再降低搜尋長度，Kademlia改良原先前置共通的特性，將一次一個bit個改為一次b個bit，如此可將搜尋長度降低至$O(\log_2{}^b(N))$。但是相對地，路由表的大小將變為$O(k*(2^b-1)*\log_2{}^b(N))$[5]

雖然 Kademlia 論文中，沒有提及如何維護二元樹，但是必花費額外的維護訊息。而且前置共通會造成許多問題：當分支再次分岔時，相關節點必須利用額外的訊息來更新其子樹集合。又因記錄子樹時沒有固定模式，當點離開時，很可能造成相對子樹遺失，簡單地說，就是暫時失去部分區域，直到下一次更新成功為止，避免的方法就是 k 加大，但是這樣將造成路由表大小倍增。而且某點同時成為多點的相關子樹代表點時，該點將成為熱點。事實上前置共通也可能造成二元樹的不平衡，增加搜尋負擔。

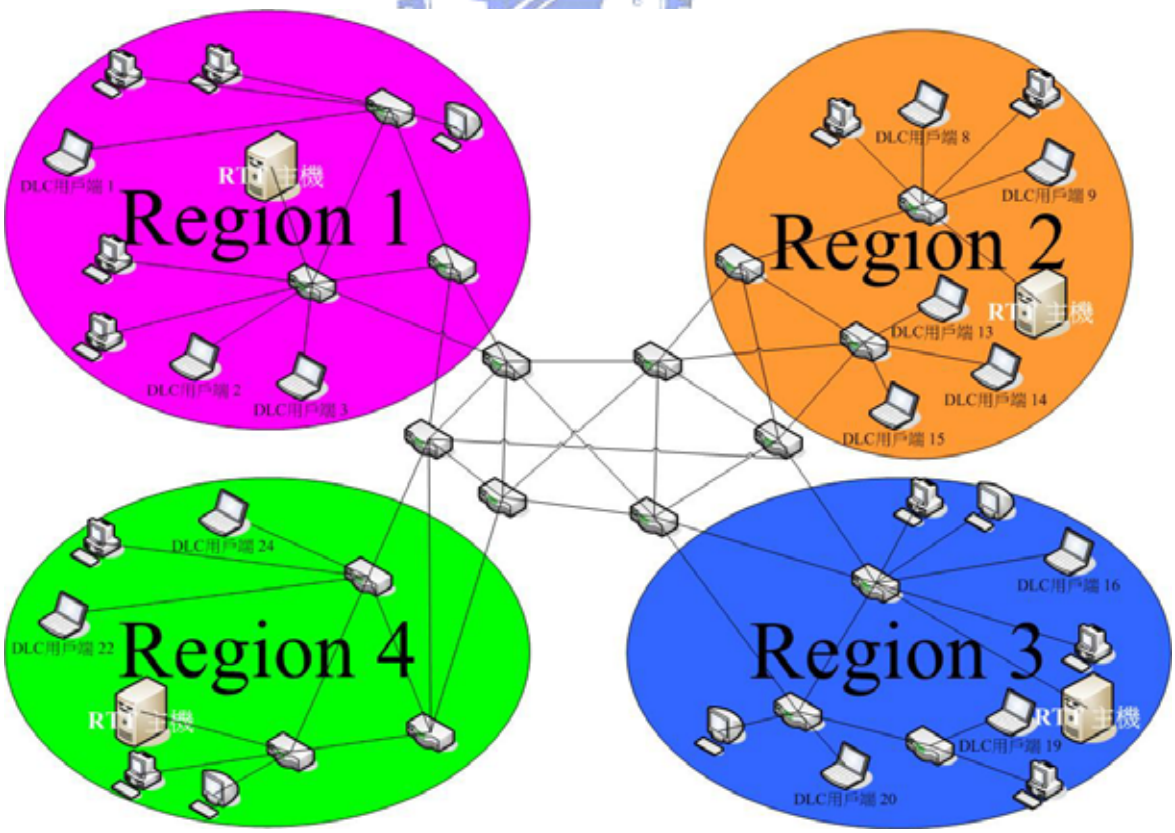


圖表 33 Kademlia 的搜尋

第四章 系統架構

DLC－Double direction Locality Chord，顧名思義就是加入了雙向搜尋、地域技術的 Chord，另外 DLC 還加上 BT 追蹤者傳輸的機制。本章將介紹點對點傳輸架構 DLC 的整體設計概念，並簡單描述各個用戶端的主要元件及其功能，最後介紹 DLC 中各程序的使用與概念。4.1 介紹 DLC 的系統概觀。4.2 講述 DLC 的設計方法。4.3 解說 DLC 用戶端軟體基本架構。4.4 描述 DLC 的主要程序流程與用途。

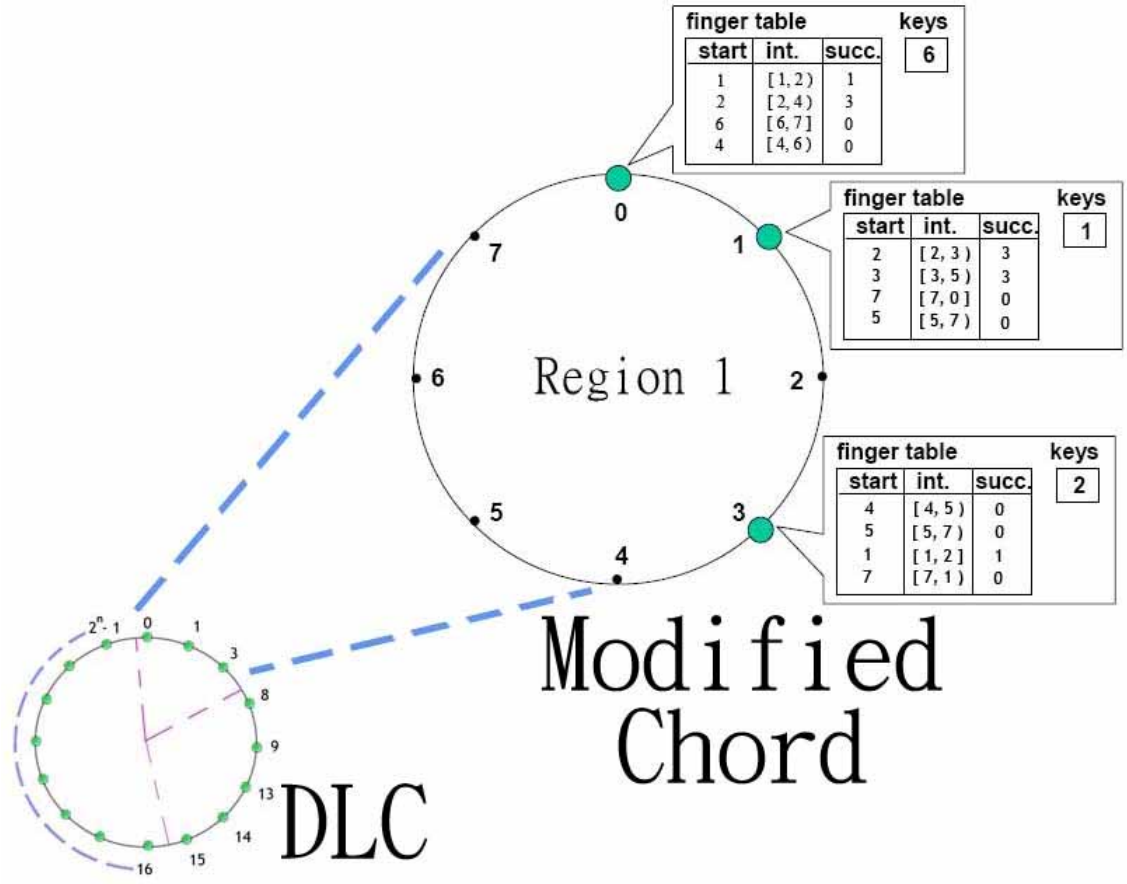
4.1 系統概觀(System Overview)



圖表 34 DLC 實體網路架構

DLC 實體網路架構中，利用多台 RTT 主機輔助將實體網路分爲多區，而同區之間的封包傳遞較爲快速，所以同區之間訊息傳遞、資料實體的傳送都較不同區間快速。利用這個特性並延伸至邏輯網路上。以整體 DLC 來說，邏輯上爲環狀網路，用戶端即爲環狀網路上的點。而環狀網路可拆成數個子邏輯網路，每個子邏輯網路即爲實體上的分區，亦爲修改後的 Chord 邏輯網路。

同子邏輯網路中的點，不僅用戶端編號接近，實體網路位置也較爲接近。圖表 34、圖表 35 中DLC的邏輯空間從 0 到 $2^n - 1$（此處的n > 3），所以一個DLC邏輯網路最多可有 2^n個用戶端同時運行，而每個分區最多可有 8 個用戶端，不過邏輯網路中通常無法達到最大用戶端數，因此會有跳號的情形，如第一分區中只有三個用戶端。每個用戶基本上都必須擁有路由表(沿用Chord的名稱finger table)，方能提供整個網路正常運行。



圖表 35 DLC 網路邏輯架構圖

4.2 設計方法與假設

DLC架構設計上除了對Chord的修改之外，更加入的雙向搜尋、主動地域代理、BT 追蹤者傳輸等功能。爲了後面解釋方便在此先設定一些參數和變數，整個DLC邏輯網路分爲八個子邏輯網路，用戶端編號空間爲 0 到 $2^n - 1$（n必須大於 3），即用戶端編號需要n個bit來代表，所以最多可有 $2^n = N$個節點同時在此邏輯網路中運行。

4.2.1 雙向搜尋(Double Direction Search)

雙向搜尋(DDS，Double Direction Search)是源自 Kademlia 提出的問題『相等編號距離應該有相等的跳躍數』。爲了維護這個效益，Kademlia 利用變形二元搜尋樹架構[5]，但是前述這樣的成本較高。所以本論文採用更簡單、更安全的辦法，使得單向的 Chord 變成雙向。

主要的概念在於增修Chord的路由表內容。DLC和Chord路由相同，有三個主要欄位：區間(interval)、區間起點(start)、後繼者(successor)。區間是指子邏輯網路的再分割空間，其上數字即爲用戶端編號，簡單地說區間就是負責範圍。區間起點就是理論上該區間的第一個用戶端編號。後繼者則是因爲有可能跳號，實際上該區間的真正負責點。雙向搜尋即是利用區間的概念，原先Chord區間爲正向$[(r+2^k) \bmod N, (r+2^{(k+1)}) \bmod N)$(r 爲自身用戶端編號，$0 <= k < \log_2 N$)，共$\log_2 N$個；而DLC正向區間$[(r+2^k) \bmod N, (r+2^{(k+1)}) \bmod N)$、逆向區間$[(r-2^k) \bmod N, (r-2^{(k+1)}) \bmod N)$各$\log_2 N - 1$ 個共 $2 * (\log_2 N - 1)$列。

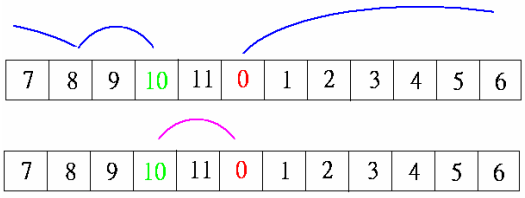
列號	Chord 區間	DLC 區間	DLC 區間(修改)
1	[1, 2)	[1, 2)	[1, 2)
2	[2, 4)	[2, 4)	[2, 4)
3	[4, 8)	[4, 8)	[4, 8)
4	[8, 16)	[8, 16)	[8, 16]
5	[16, 0)	[31, 30)	[31, 30)
6	X	[30, 28)	[30, 28)
7	X	[28, 24)	[28, 24)
8	X	[24, 16)	[24, 16)

表格 3 Chord 與 DLC 的路由表區間

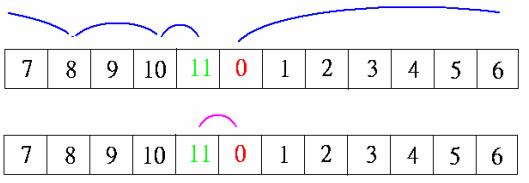
以N = 32，r = 0 爲例，如表格 3。Chord的區間範圍爲$[(r+2^k) \bmod N, (r+2^{(k+1)}) \bmod N)$

[3]，而 32 對 32 的餘數爲 0，32 即爲點本身，所以原先Chord的區間可以完整覆蓋整個邏輯網路從 0 到 31。如果前數大於後數，如表格 35 中Chord的最後一個區間[16, 0)，表示爲環狀網路頭尾相接處。但是DLC如果直接套用區間公式，將會遺失一個點r = 16，如表格 3 的DLC區間所示，爲了改善此問題，將正向的最後區間範圍改爲[$(r+2^{k})$mod N,$(r+2^{(k+1)})$mod N]。

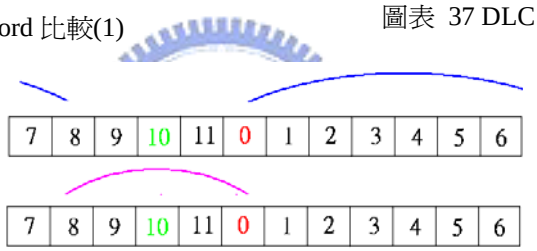
搜尋時，雙向搜尋可有兩方向選擇，但用戶端可以輕易判斷正向跳躍或是逆向跳躍較佳，進而選擇一個方向。在非特殊例子中如圖表 38，跳躍數最多不會超過原先的Chord。但是在部分例子中如圖表 36、圖表 37 則可將低跳躍數。



圖表 36 DLC 與 Chord 比較(1)



圖表 37 DLC 與 Chord 比較(2)



圖表 38 DLC 與 Chord 比較(3)

我們模擬實驗CAN、Torus CAN、Chord、DDS四種搜尋跳躍數比較，實驗中共有2^{18}個節點填滿整個邏輯網路，執行數萬次亂數取點搜尋，分別在一維、二維、三維中的表現，其結果如表格 1。

	1D	2D	3D
CAN	86922.44	334.039	63.972
Torus CAN	67162.251	261.306	48.250
Chord	9.023	8.962	8.887
DDS	6.185	6.440	6.665

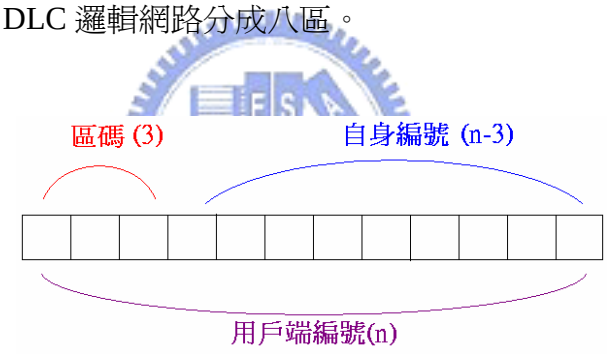
表格 4 CAN、Torus CAN、Chord、Double Chord 搜尋跳躍比較

很明顯地，Chord 與 DDS 的搜尋跳躍比 CAN、Torus CAN 少許多。而 Chord 與 DDS 之間以 DDS 較佳。不過這算是 Trade off，因爲 DDS 需要的維護訊息將大於 Chord。而

實際上 DDS 降低的幅度，並不像圖表 36 圖表 37 那麼明顯，因爲特殊情況多是出現於目標位置於逆向且相當靠近原先發出搜尋的點，所以平均下來降低有限。實際網路環境下點與點之間更爲分散，效果將更加有限。不過論文中，我們還是以 DDS 爲搜尋設計主體，因爲這樣的機制相當於有雙重保障，當較佳搜尋無法進行，亦可採用逆向導正，降低搜尋失誤率。

4.2.2 主動地域代理(Active Locality Proxy)

網際網路常以代理技術來增加傳輸、降低熱點負載。一般來說，代理機器越多平均負載越低，也越能提升搜尋成功率。但是代理機器多，有效性與同步問題也相對提昇，而最大問題在於代理機器的分佈問題，因此 DLC 導入地域的觀念，至於要分爲幾區，CAN 論文指出二維四區的效能是合理且節省的，而事實上 CAN 的二維可視爲兩層一維重疊起來[1]。基於這個理論 DLC 採一維八區設計，所以系統亦假設網際網路中均勻分佈八台地標主機，將 DLC 邏輯網路分成八區。



圖表 39 用戶端編號

爲了使邏輯分區趨近實體分區，DLC對用戶端編號做修改。利用二進位的特性：MSB(most significant bit)有較大權重，又因分爲 $2^3 = 8$，所以前三個MSB代表區碼。其他 n – 3 個bit就是利用雜湊函式算出的自身編號，修改後的用戶端編號如圖表 39 所示。

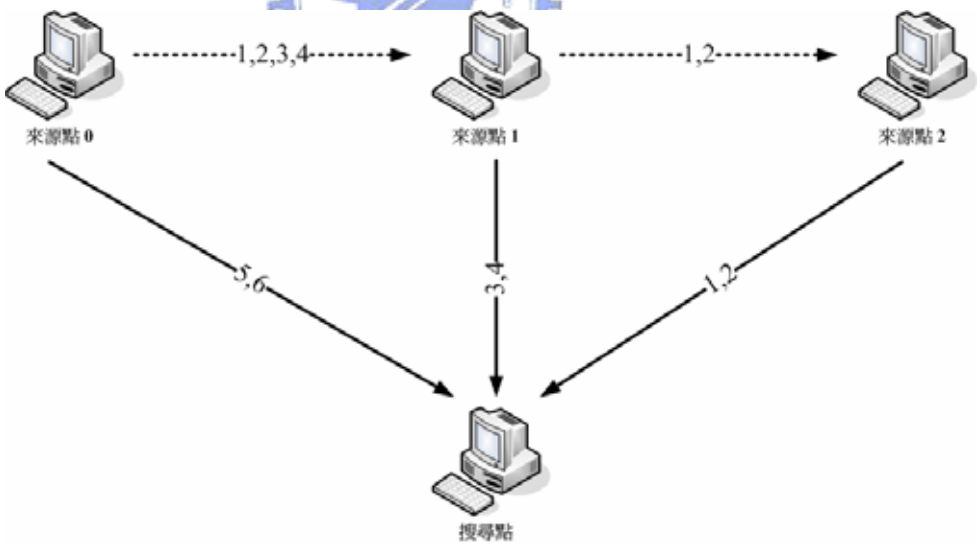
DLC 分區後每個區域都可以獨立視爲一個 Chord。而分享資料時，除了將檔案資訊送到同區域負責的用戶端之外，還必須主動推送到其他區域的負責用戶端上，而負責維護同一檔案的用戶端其自身編號皆相同。資料必須放在自身編號爲 N 的點上，所以八個自身編號爲 N 的點都會擁有該資料的媒體檔案，當然每區域中不一定會有自身編號爲 N 的點，所以我們會直接採用後繼者來輔助。所以搜尋資料時，只需對所屬的分區做搜尋即可。這樣不僅可分散搜尋熱點負載達到代理的目的、亦可降低原始 Chord 中因一點遺失而造成與該點相關的媒體檔案全部喪失的機率。至於代理的時效性，本論文建議採用

逾時取消，因爲分爲八區後，每點可能要承受八倍的媒體檔案資料量，如果額外花成本來維護代理機制，將浪費過多網路成本。但是爲了確保媒體檔案的存活，代理點可在搜尋失敗時，主動向擁有相同自身編號的點做查詢，確定整個 DLC 都沒有該搜尋目標才真正回傳搜尋失敗訊息給發出搜尋點。

在不管法律的限制下，負責的用戶端更可提升爲檔案實體代理，於網路使用頻率較低的時段，利用主動傳輸來取得熱門檔案實體，藉以提升傳輸品質。此外，亦可升高檔案在整體網路的存活時間。不過這也得考慮用戶端的容量、穩定性等因素，不過本論文中皆假設用戶端同意且合法成爲檔案實體代理。

4.2.3 BT 追蹤者傳輸(BT Tracker Transmission)

除了利用地域特性加快傳輸之外，同時也採用目前公認最快的傳輸方式－BT 追蹤者的概念。作法很簡單，前述主動地域代理中，每點必須維護與自身用戶端編號相關的資料外，同時也爲該些資料的追蹤者，負責同區中的資料傳輸輔助，藉由這項改進，解決了原先 BT 必須尋求穩定追蹤者的缺點，而且也減輕身爲追蹤者的負載。



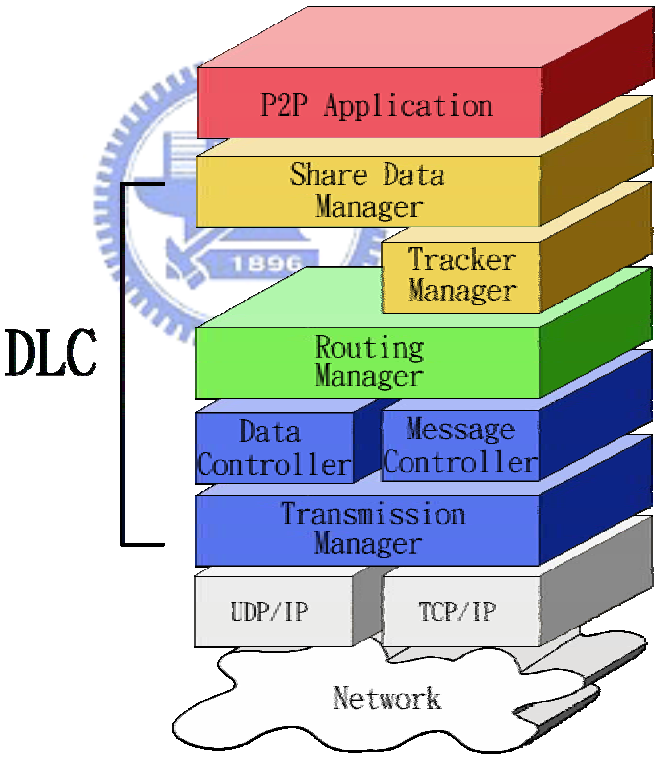
圖表 40 BT 追蹤者傳輸

第一次傳輸，除非本身擁有檔案實體，否則檔案必由他處傳至搜尋點，當用戶端經傳輸得到檔案後，追蹤者會記錄該用戶端擁有哪些檔案片段。當該資料再次被要求傳輸時，追蹤者會盡可能以同區來源點爲回傳結果，如圖表 40 所示，先前來源點 1 向來源點 0 取得資料片段 1、2、3、4，來源點 2 從來源點 1 取得資料片段 1、2，而最新的搜

尋點可同時從三個來源點取得資料片段。所以 BT 追蹤者傳輸可使越熱門的檔案擁有越多來源，藉此提升傳輸速度，同時也提升檔案實體的存活率。

4.3 用戶端軟體架構

DLC 屬於第三代點對點架構，每點的重要性與功能皆相同，所以每點的軟體架構也都相同。軟體架構設計上，DLC 介於應用層與 TCP/IP 之間，分爲四個主要管理元件，包括控制所有傳輸的傳輸管理元件(Transmission Manager)、負責繞送路徑的路由管理元件(Routing Manager)、資料追蹤管理元件(Tracker Manager)、分享檔案管理元件(Share Data Manager)等。藉由四個管理元件合作，即可維護一個用戶端的行爲。下面將一一詳細介紹各管理元件的作用，與其他管理元件的關係。



圖表 41 DLC 用戶軟體架構

4.3.1 傳輸管理元件(Transmission Manager)

傳輸管理元件是建構在 TCP/IP 之上，負責 DLC 的所有傳輸。而一個用戶端的所有相關傳輸，又可以分爲溝通訊息與資料傳輸兩類。所以架構設計上，也分爲兩個子控制元件，以便清楚劃分責任歸屬。

溝通訊息由訊息控制元件(Message Controller)來負責收發。訊息控制元件通常接收到的訊息爲邏輯架構維護訊息或者媒體資料更新訊息兩種。接收到邏輯架構維護訊息時，該控制元件會將訊息分析後，回傳至路由管理元件進行路由表更新的動作。接收到媒體資料更新訊息時，會傳至資料追蹤管理元件。另外，自身發出的邏輯架構維護、媒體資料更新訊息，都是通過本元件來發送。

真正資料傳輸則是由資料傳輸控制元件(Data Transmission Controller)來負責。上傳資料時，該元件必須負責所有資料實體傳輸要求與分享檔案管理元件之間的溝通，如傳輸某資料的檔案片段給哪個點。下載資料時，該元件也必須管理整合已下載資料片段、對毀損片段主動重傳或修復等。傳輸管理元件除了資料傳輸之外，也提供附加傳輸功能如資料安全的加密解密、資料來源的傳輸品質檢測，也必須負責與傳輸相關的低階溝通訊息，如傳輸埠(port)選擇等。

4.3.2 路由管理元件(Routing Manager)

資料搜尋是點對點傳輸架構中最重要的功能。資料搜尋就是查詢封包的轉送，而轉送的優劣與否，往往就是資料搜尋的評比關鍵。爲了快速搜尋，幾乎所有點對點傳輸架構都採用路由表(Routing Table)輔助封包轉送的概念。所以 DLC 也不例外，並將此功能獨立成一個路由管理元件。路由表更新主要分爲：心跳監測機制(heartbeat monitor)、外部傳來的邏輯更新訊息兩種情況。心跳監測機制是爲了維持邏輯網路的正常運行，每隔一段時間，點就檢查路由表中的後繼者是否存在，後繼者回傳時，表示存在之外，亦可能通知有更適合可爲後繼者的點，進而觸發一連串的更新。當然，如果後繼者已經離開邏輯網路，那麼將沒有回傳資訊，經過多次測試後，發現後繼者確實離開後，該點就會自行利用其他後繼者，找到應有的後繼者來代替，然後開始觸發相關點的更新訊息。外部傳來的邏輯更新訊息就是其他點心跳監測後產生的觸發更新。

更新觸發後將產生一連串的更新程序，Chord中證明只需$(\log_2 N)^2$則更新訊息[3]，而DLC因DDS且需與其他地域的對應點溝通，所以需要$(2*(\log_2(N/8)-1)^2+7)$則更新訊息，最差情況下約爲Chord更新數量的兩倍。

4.3.3 資料追蹤管理元件(Tracker Manager)

因爲點本身爲檔案追蹤者，所以追蹤者的工作就交由資料追蹤管理元件來處理。資

料追蹤管理元件處理的檔案，只限於本身管理的資料，即資料編號對網路空間的餘數等於自身編號者。

點加入 DLC 後，會有媒體資料和查詢封包傳來。當搜尋到本身擁有的媒體資料時，除了回應何處有資料實體之外，資料追蹤管理元件也會持續與發出詢問的點溝通，並獲知該點能否成爲新資料來源，如果可以則新資料來源也會被更新到媒體資料。此外，資料追蹤管理元件每隔一段時間發出更新媒體檔案給各區的對應點，藉以做到代理機制。實際上會出現搜尋比媒體資料先到的可能，畢竟媒體資料是陸續傳來的，其實是可以廣域搜尋的方法詢問各區域的負責點，不過基於已有主動代理的機制，且實際上搜尋失敗後使用者多會重複搜尋的理由，本論文採用最簡單的方式－同區域內，搜尋比媒體資料先到視爲搜尋失敗。

資料追蹤管理元件負責維護媒體資料，然而媒體資料可提供的相關資訊非常多樣化，但也沒有統一的規格。基於這個理由以及往後擴充方便，本論文建議以 XML Scheme 配合 XML 的格式來儲存。此外，因爲 XML 可以透過 XSL 做資料轉換，所以其他應用軟體只要能夠解析 XML Scheme 與 XML，並遵守 DLC 的傳輸規定，亦能輕易加入 DLC。藉由不同點對點傳輸平台的互相溝通，使得資源分享的可行性更高。

對於媒體資料的時效性，建議採用與代理機制相同的逾時取消，因爲熱門資料多是有時效性的，很少有資料可以有長時間的高搜尋、高下載頻率，否則過多使用頻率低的媒體檔將會佔據過多儲存空間，不僅增加用戶端的負擔也會增加網路成本。此外，媒體檔案中來源點的數量可能相當多，爲了方便與公平起見，建議以最後接觸(Access)的幾點，即最後有資料實體傳輸的幾點爲來源點。實際上，當然也要考慮完整資料的可取得性，不過本論文中，暫不考慮此一問題。

4.3.4 分享檔案管理元件(Share Data Manager)

分享檔案管理元件用以管理本身要分享的檔案，主要工作就是產生資料媒體，並送到各區域負責點上，進而提供整個邏輯網路搜尋，雖然會有些許延遲，但足以稱爲廣域搜尋，這也是本論文宣稱 DLC 能以區域搜尋做到廣域搜尋的原因。

有鑑於一般點對點傳輸常出現下載之後無法使用的問題，分享檔案管理元件也會檢查本身分享的資料是否完整，如果尚未完整，分享檔案管理元件會主動與訊息控制元件溝通搜尋毀損或短缺的資料片段，並進行重新下載，避免其他點下載後無法使用。此外，

先前提及本論文暫不考慮合法性的問題，加入了檔案實體代理，所以分享檔案管理元件會主動推送檔案實體到各區負責點上，藉此提升檔案存活以及稀少資料的散佈。

分享檔案管理元件做資料分享時，和一般的點對點應用軟體相同，也是以切割資料的概念來處理傳輸檔案，系統提供一個基本容量單位，所有大於基本容量單位的資料都會被切割成數個資料片段。舉例來說基本容量爲 4MB，則一個容量爲 17MB 的資料，將會被切割成 4 個 4MB 與 1 個 1MB 的資料片段。某點只要擁有完整的資料片段，即可成爲資料來源點。實際上分享檔案管理元件可掛載於檔案系統之下。所以對使用者來說，下載後的資料無論是完整資料或者資料片段都會放置在某資料夾中，而使用者可以直接對該資料夾做存取，藉以增加或減少分享資料的數量。

4.4 主要程序訊息

點對點傳輸架構的運行，主要是由更新、加入、離開、查詢、分享組成，DLC 是建立在 Chord 之上，並加入額外的邏輯概念，所以 DLC 的程序訊息是可行且穩定的。後面小節會簡略步驟。

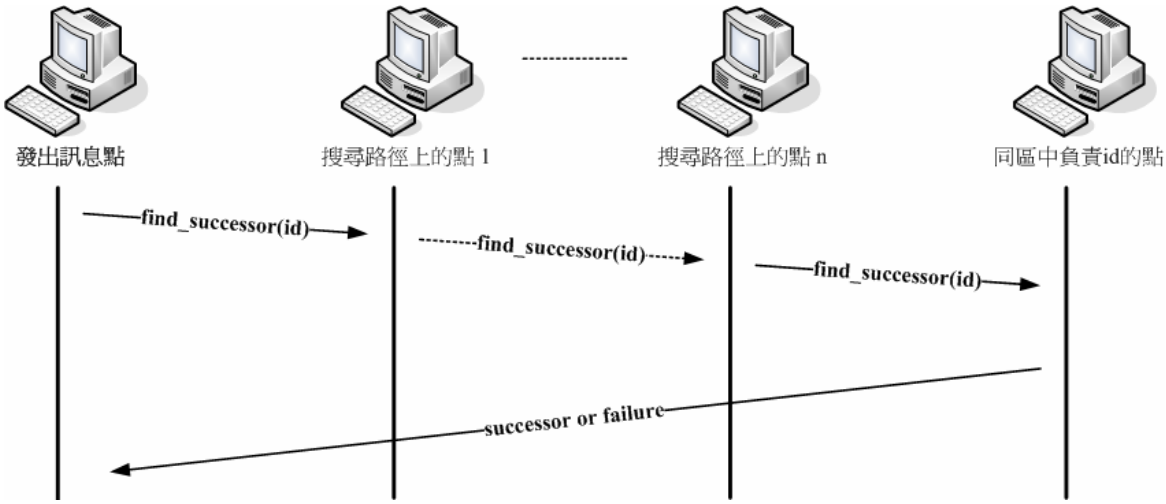
Notation	Definition
finger[k].start	(n + 2^{k-1}) mod 2^{m}，1 <= k <= m
.node	First node >= n. finger[k].start
successor	The next node on the identifier node ; finger[1].node
predecessor	The previous node on the identifier circle

表格 5 程式碼詞彙表

DLC 的程序訊息主要分爲五種：查詢、更新、分享、加入、離開等。下面各小節將一一說明各程序訊息的功能與行爲。

4.4.1 查詢(Look Up)

查詢是點對點網路中尋找分享資源的方法，基本上先找出後繼者，進而對後繼者做真正的查詢動作。Chord 與 DLC 的不同點在於 Chord 尋找找整個邏輯網路中的後繼者，而 DLC 尋找同區中的後繼者。



圖表 42 DLC 尋找後繼者(find_successor)

DLC 查詢前，點必須找出尋找目標的關鍵值，最常見的就是檔名，利用關鍵值計算出媒體資訊應該在哪一個用戶編號，進而對照路由表，找出所屬區域中的後繼者。查詢的封包會一直被轉送，最終到達到該區負責點。最大轉送數爲前述的 log(N/8)。傳輸通常開始於查詢之後，利用查詢到的檔案來源下載檔案實體。

另外，查詢的機制在 Chord 與 DLC 也都扮演輔助加入、更新程序的角色，差別在於查詢是在於尋求媒體資料，而加入、更新在於尋求可用的點成爲後繼者。

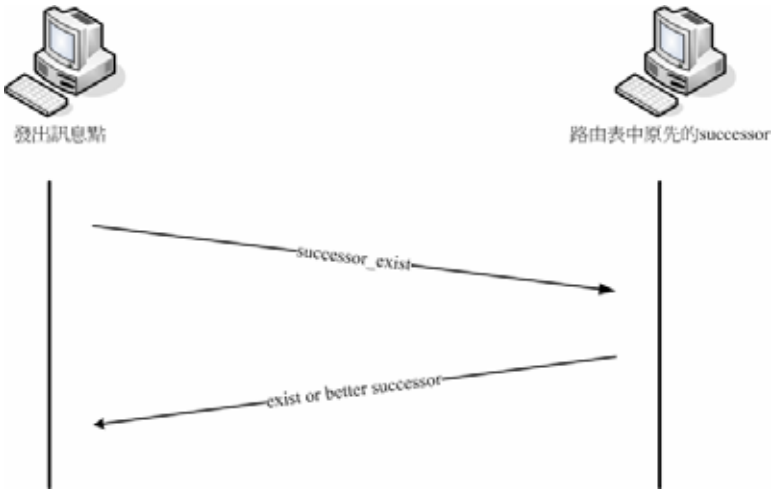
4.4.2 更新(Update)

更新在整個邏輯網路中，主要的功能就是維護邏輯網路中各點的順序關係，以及點與後繼者的關係，否則搜尋資料時可能需要走更遠的路徑，甚至錯誤的路徑。更新程序訊息主要就是維護路由表的正確性，而更新訊息的產生主是定時維護的機制。

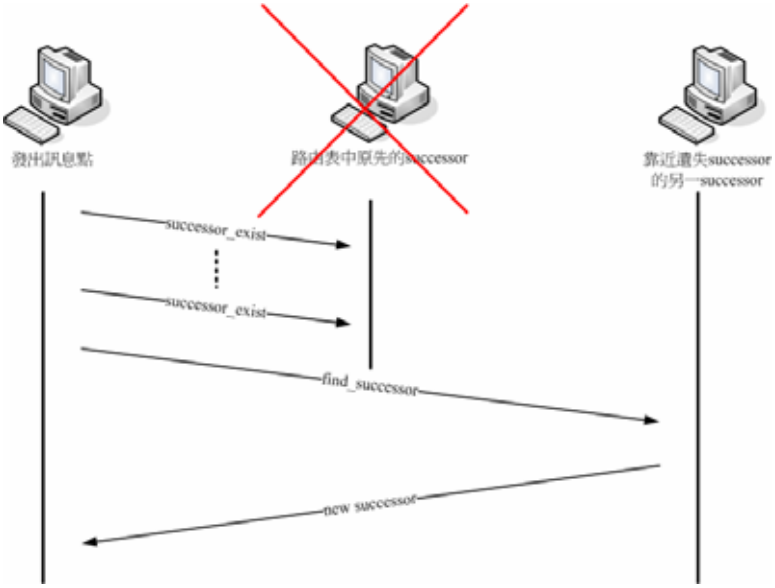
定期維護是指，每一時間間隔，點就發出監測訊息給路由表中的各個後繼者，後繼者回傳訊息又可分爲兩種情形：正常更新與非正常更新。

正常更新如圖表 43，回傳後繼者存在(exist)表示該原先後繼者可繼續勝任，或者回傳更適當的點可代替原先的後繼者，此時發出訊息點將會更改路由表。

非正常更新圖表 44，如果受監測一方持續沒有回應，多次之後發送訊息點將會視該後繼者已經離開邏輯網路，此時，發送訊息點會發出搜尋後繼者的訊息給靠近遺失後繼者的其他後繼者，最終發出訊息點會得到一個新的後繼者。



圖表 43 正常更新

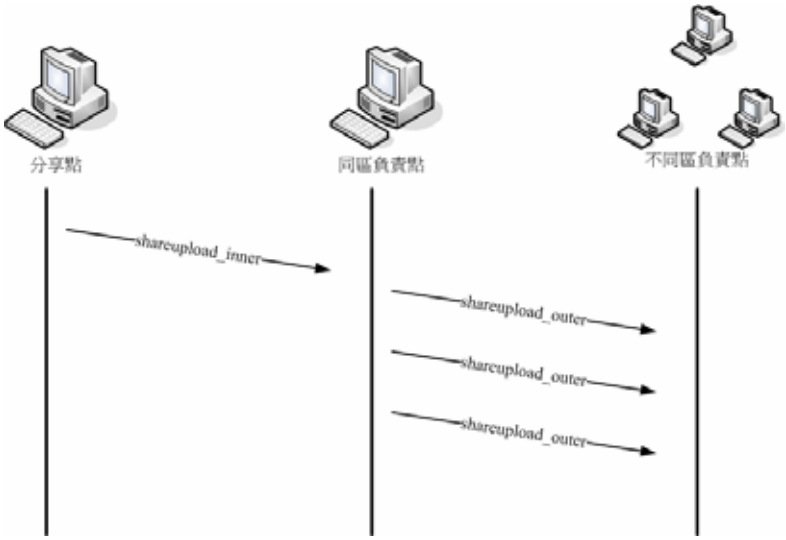


圖表 44 非正常更新

4.4.3 分享(Share)

分享是指，新點加入邏輯網路時主動分享自身擁有的資料給整個邏輯網路，或者點離開邏輯網路時，其原先維護的媒體資料必須轉由其他點來維護，此處的分享可以是資料實體、亦可是媒體資料，視實際執行環境而定。

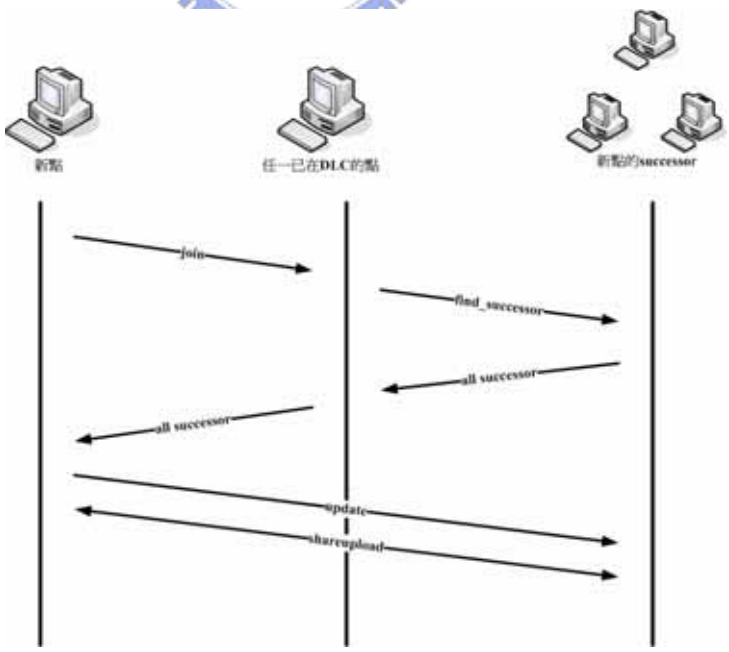
分享訊息程序又可分爲兩種分享：同區分享(shareupload_inner) 與非同區分享(shareupload_outer)。同區分享是指，檔案實體擁有者主動將檔案實體的媒體資料傳至同區中負責該檔案實體的點上；非同區分享是指，同區負責該檔案實體的點，主動將媒體資料傳送到其他區域中的負責點。



表格 6 分享程序訊息

4.4.4 加入(Join)

加入是指一個新點加入邏輯網路。加入 DLC 必須透過一個已在 DLC 邏輯網路中的點，藉由該點提供加入點的所有後繼者資訊，之後新點會發送更新訊息給所有後繼者，而這個更新訊息將會導致一連串的更新，最後傳送分享媒體資訊。如果新點無法得到任何點的輔助，該新點就自己形成一個 DLC 網路，DLC 邏輯網路最初即如此。

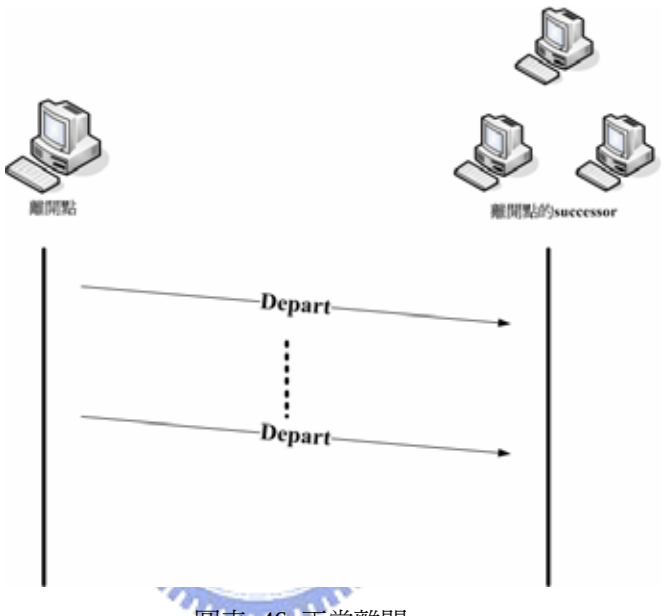


圖表 45 加入

4.4.5 離開(Depart)

離開是指用戶端邏輯上離開整個點對點網路。其中又可分爲兩種離開，一是點正常離開，二是點非正常離開。

正常離開是指，點依照正常的程序通知所有相關點，觸發更新直到所有相關點找到新的後繼者，並確保安全關閉與其相關的所有傳輸後，該點才真正離開 DLC 邏輯網路。如果所有點都能以正常離開程序離開邏輯網路，整體品質必有較好的表現。



圖表 46 正常離開

非正常離開是包含所有正常離開以外的情形，通常是用戶端斷電、故障，或者該地區網路癱瘓等情況。發現非正常離開，通常是由其他用戶端以多次心跳監測後所得的結論，進而發出更新。不過論文中都預設點的離開爲非正常離開，藉以增加誤差。

第五章模擬結果

這一個章節中利用模擬器來驗證 DLC 對於 Chord 的效能改進。本論文採用自行設計的模擬器-P2PNS 來模擬 P2P 架構的行爲，P2PNS 可同時進行多項測試評比。評量指標主要分爲搜尋、傳輸、負載等三方向，各方向中又細分成幾個實驗，基本上各實驗皆可提出實質的效能應對。5.1 介紹 P2PNS 架構。5.2 解釋評量的相關名詞及其代表的實質意義。5.3 介紹模擬實驗中的軟硬體平台，並說明測試數據與結果。

5.1 P2PNS 簡介

P2PNS(Peer-to-Peer Network Simulation)是本論文用來作爲測試的模擬平台，主要由點對點網路框架(P2P Network Framework)、行爲產生器(Action Generator)與計時器(Clock)三者組成。設計不同的點對點網路必須繼承點對點網路框架並做修改。行爲產生器則是負責合理產生各種網路行爲，藉以代表真實世界中的應用。計時器主要負責整個模擬環境的虛擬時間同步，計時器的一時間單位往往是實際時間的幾百甚至幾千倍，端視模擬網路行爲多寡而定。下面將介紹 P2PNS 的兩個主要元件：點對點網路框架、行爲產生器。

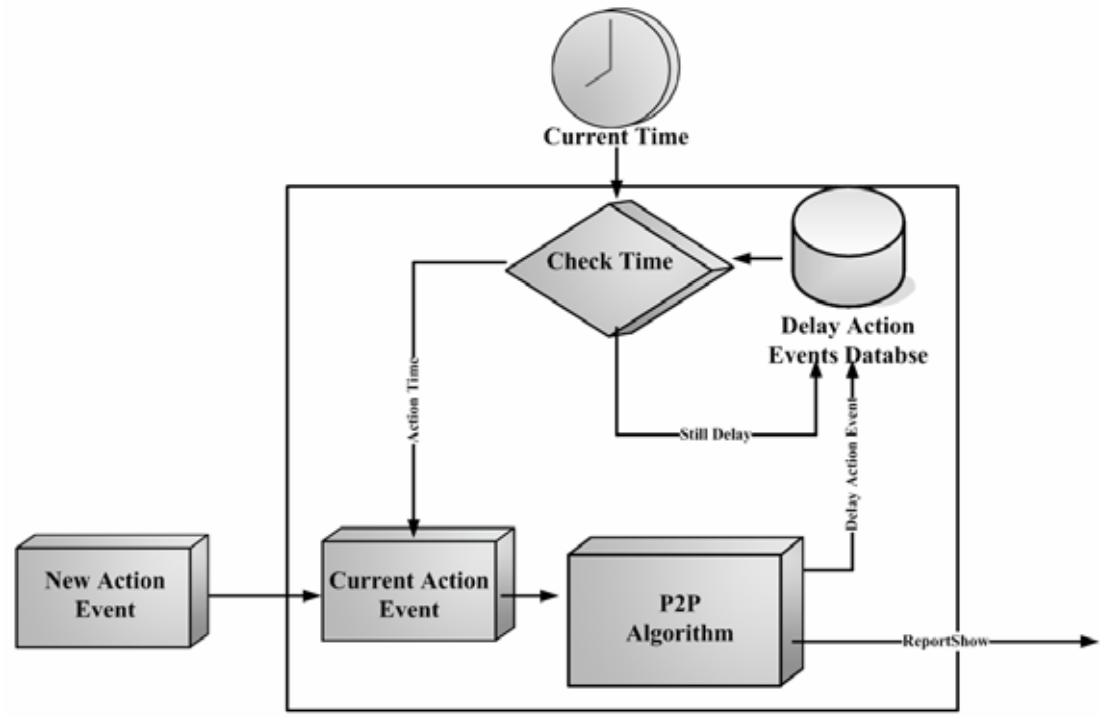
點對點網路框架(P2P Network Framework)

點對點網路框架概觀如圖表 47，點對點演算法(P2P Algorithm)是整個框架的核心，因爲所有的模擬行爲都是在其內運行包含用戶端的搜尋、傳輸以及實體網路設備的使用。繼承框架後主要的工作是利用點對點演算法提供的 API 重新實作加入、離開、分享、搜尋等行爲事件。至於用戶端與網路設備上的統計數據，API 底層的函式會主動累積計算。

整體而言，點對點網路框架是建立在時間軸上，所有的行爲事件都有其有效期限。現行行爲事件(Current Action Event)就是某一瞬間所有必須執行的行爲事件集合。新的

行爲事件都是由外部的新行爲事件(New Action Event)傳入現行行爲事件。除此之外，行爲事件常常會有延遲，如網路壅塞產生的延遲、封包傳遞的延遲等，這些有時差的行事件都是利用延遲行爲事件資料庫(Delay Action Event Database)來輔助。而所有的延遲行爲事件會經由驗證時間(Check Time)的機制，判斷是否再次加入現行行爲事件，或者因逾時而丟棄。

使用者取樣只需暫停現行時間，然後命令點對點網路框架對當時網路作統計即可，送出的結果即爲各項評比的相關數據。

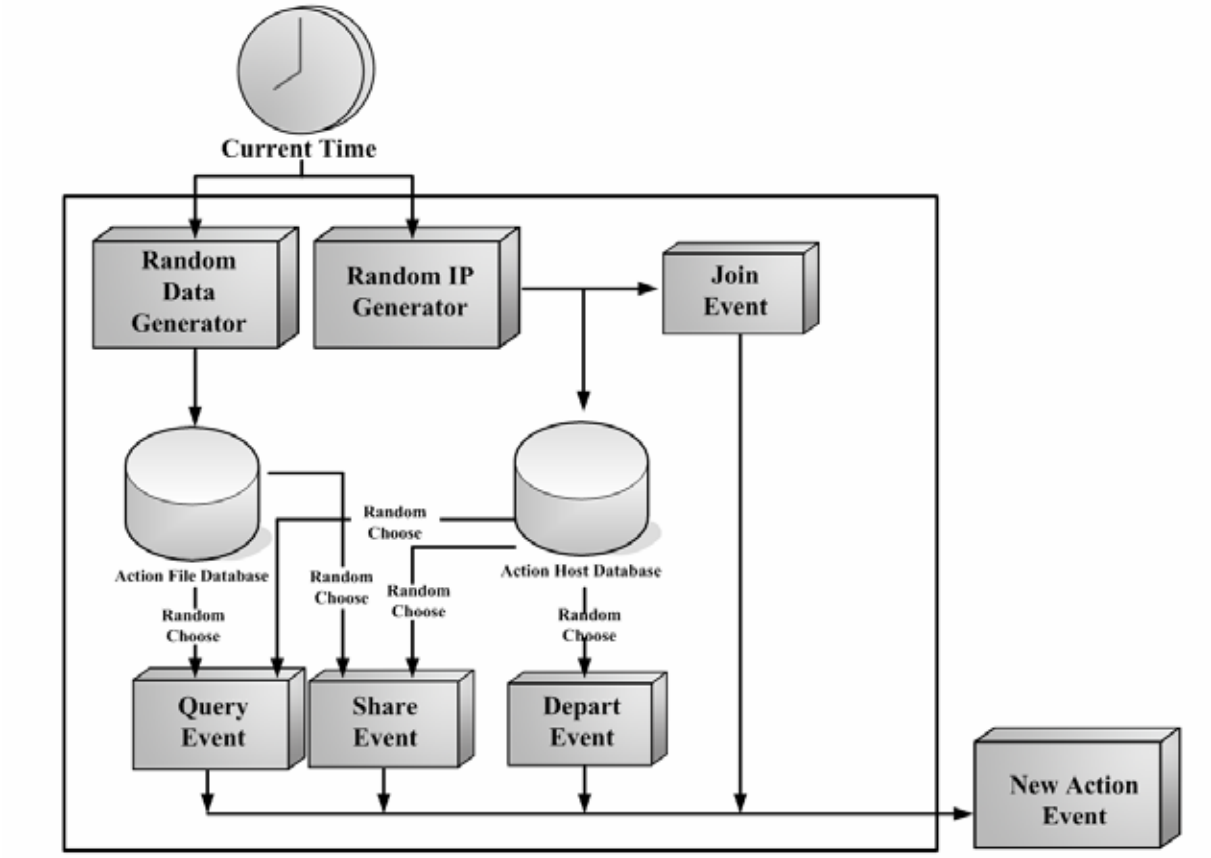


圖表 47 P2P Network Framework

行爲產生器(Action Generator)

行爲產生器主要的功用就是產生合理的加入、離開、分享、搜尋等四個行爲事件。所有用戶端資訊與所有資料都是亂數產生的，整個邏輯網路的運行單位是用戶端，但是實際上只是一個 IP，因此 IP 亂數產生器(Random IP Generator)即爲產生用戶端的機制。資料亂數產生器(Random Data Generator)則是用以產生資料及其相關資訊，如檔案大小，檔案編號等。產生之後，可用用戶端和資料各記錄在運行用戶端資料庫(Action Host Database)與運行檔案資料庫(Action File Database)內。

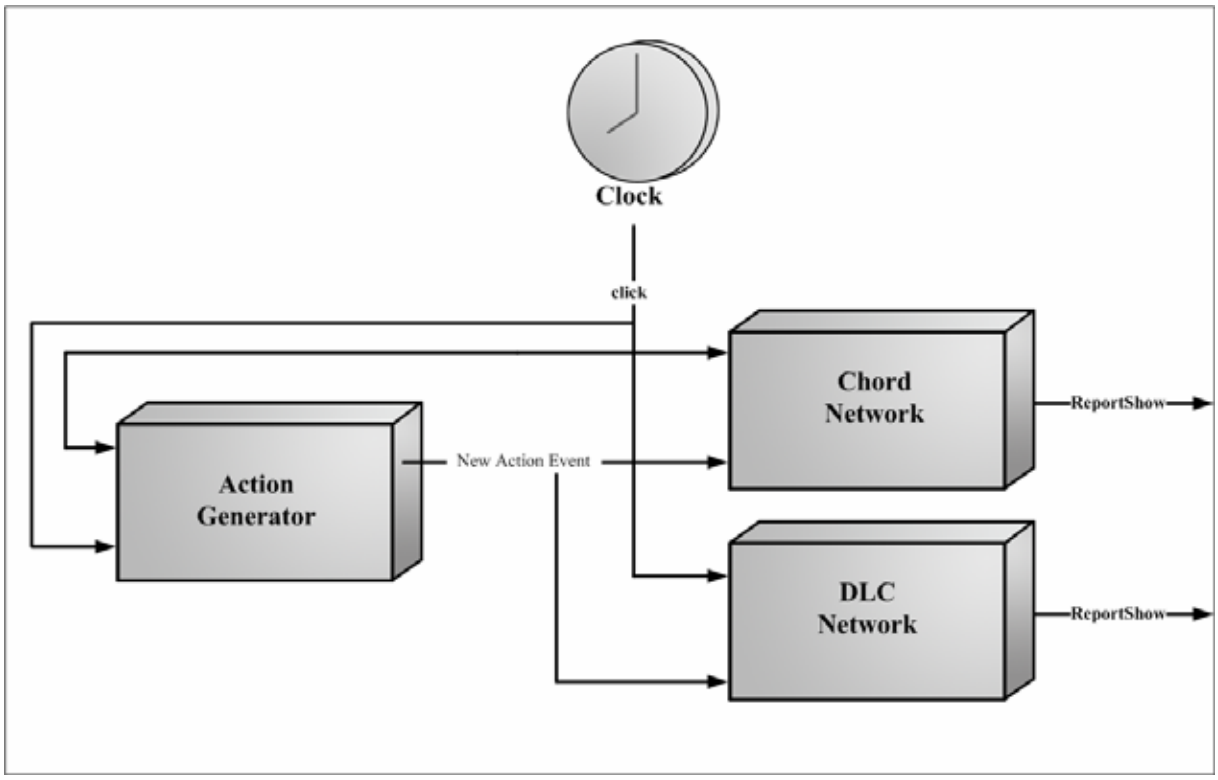
加入事件(Join Event)直接來自 IP 亂數產生器產生的合理 IP，所謂合理就是不能產生的 IP 不與運行用戶端資料庫中任一 IP 重複。離開事件(Depart Event)則是自運行用戶端資料庫中選出數個用戶端成爲離開的用戶端。而分享事件(Share Event)和搜尋事件(Query Event)都是同時配合運行用戶端資料庫與運行檔案資料庫產生的。每一個單位時間，四者皆產生數個不同的事件，而所有產生的事件合稱新行爲事件。



圖表 48 Action Generator

P2PNE 概觀

模擬實驗同時比較 DLC 與 Chord 兩種點對點網路，因此本論文使用的模擬器系統概觀如圖表 49，其中 Chord Network 與 DLC Network 都是繼承點對點網路框架而實作的。爲了公平起見，兩者採用相同的計時器，所執行的行爲事件也完全相同，藉此將其他變數產生的誤差降到最低。每當計時器增加一個時間單位，行爲產生器將產生新行爲事件，複製兩份同時傳送給 Chord Network 與 DLC Network。使用者可在任意時間監測兩種網路的行爲及統計數據。



圖表 49 P2PNS 系統概觀

P2PNS 開發平台：

JAVA：j2sdk1.4.2_07

IDE：Eclipse Platform Version: 3.1.0

設計 P2PNS 時，考慮模擬時間可能相當漫長，爲了同時在多台機器上執行以減低模擬時間，所以採用 Java 來設計以便能在各種不同的機器上執行，此外 Java 提供的 API 也較爲豐富，減少許多開發時間的浪費。

5.2評量指標(Criteria)

評量指標分爲搜尋、傳輸、負載等三方向，依照各方向特性再選出具有代表意義的評量項目。搜尋方面以搜尋成功率、搜尋路徑長度爲評量項目，傳輸方面以傳輸成功率、傳輸進度爲評量項目，負載方面以用戶端搜尋負載、路由器傳輸負載爲評量項目。下面將說明各評量項目的定義與實際意義。

搜尋成功率(search success rate)：搜尋成功率，即某時距內搜尋成功總數除以搜尋總數。數值越高，表示越容易找到傳輸來源。搜尋成功率實際代表用戶端搜尋到媒體資料的機率。

搜尋路徑長度(search path length)：搜尋路徑長度，即某時距內所有搜尋的搜尋路徑長度除以搜尋總數，此數值越高表示搜尋轉送次數越多。一般來說，搜尋路徑長度越長，搜尋時間也會越長。搜尋路徑長度實際可代表用戶端等待搜尋回應的時間。

傳輸成功率(transmission success rate)：傳輸成功率，即某時距內傳輸成功總數除以傳輸總數，此數值越高表示透過傳輸取得完整檔案的機會越高。此處特別獨立出傳輸成功率，是因爲前述的搜尋成功率並不能代表傳輸必定成功。

傳輸進度(transmission progress)：傳輸進度，即某時距中所有傳輸資料的完整程度，以百分比來表示。此數值越高，表示總體傳輸越完整。傳輸進度爲 0，表示所有傳輸皆無取得任何實體片段；傳輸進度爲 100，表示所有傳輸皆取得完整實體。正常來說，此數值會介於 0~100 之間。真實情況下可間接表示傳輸速度。

路由器傳輸負載(router transmission loading)：整體網路成本最簡單的測量方法就是計算網路中所有路由器的負載。而路由器傳輸負載即某時距內，所有路由器經過的傳輸用封包數量平均值。實際上該數值越高，網路設備的負載越重。

用戶端搜尋負載(host search loading)：用戶端最常做的行爲即封包轉送，轉送越多相對地用戶端的負載越重。用戶端搜尋負載即某時距內，所有用戶端被查詢的平均次數。此數值越高，表示用戶端被搜尋的頻率越高。

5.3 系統效能 (Performance Evaluation)

實際執行系統測試的模擬環境、模擬方法與硬體平台如下：

模擬環境與模擬方法：

模擬實驗中所用的硬體平台皆因爲記憶體有限，而無法模擬超過 1024 個用戶端，因此模擬時將邏輯編號空間縮小爲 2^{14}，資料編號、用戶端編號長度也改爲 14bit，避免過多的跳號造成影響。測試時，分別以用戶端平均個數爲 128、256、…、1024 爲測量標準，並在整體網路處於穩定時執行取樣。所得的數值皆爲至少測試 20 次以上的平均值。

模擬程序訊息爲系統所提供的四個主要程序訊息：加入、離開、分享、搜尋，模擬時由行爲產生器動態產生，因此每次的測試輸入皆不相同，藉以提升測試的公正性與普遍性。另外產生事件的機率區間亦是預先給定的，而分享檔案的熱門與否則是亂數給定，越熱門的分享檔案被要求搜尋與傳輸亦越多。至於數據統計如搜尋總數、搜尋成功總數等，都是由 P2PNS 中 Chord Network 與 DLC Network 取得。

硬體平台：

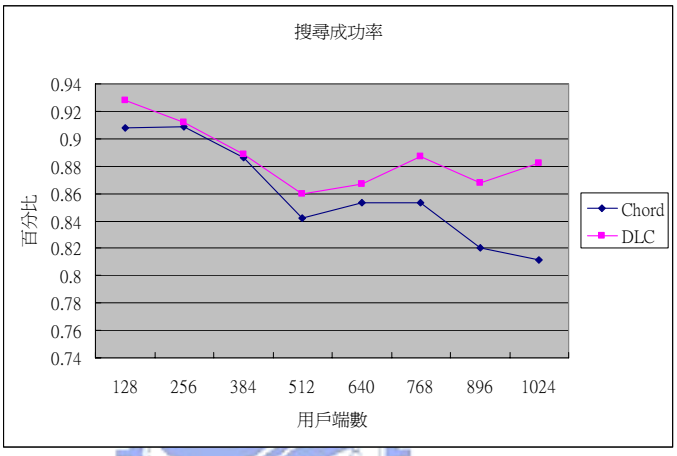
系統模擬執行時，耗費相當多時間，爲了提升數據品質，我們利用多台設備來做測試並重複數次以求平均。下面爲各硬體規格。

Name	CPU	Ram	OS
Personal	Intel Pentium 4 2.8 (HT)	1 G	Windows XP SP2
Linux2	AMD Athlon(TM) XP 2600+	2GB	RedHat Linux 9.0 (Kernel 2.4.27)
Linux3	AMD Athlon(TM) XP 2600+	512MB	RedHat Linux 9.0 (Kernel 2.4.27)
Linux4	AMD Athlon(TM) XP 2600+	2GB	RedHat Linux 9.0 (Kernel 2.4.27)
Linux5	AMD Athlon(TM) XP 2600+	1GB	RedHat Linux 9.0 (Kernel 2.4.27)
Linux6	AMD Athlon(TM) XP 2600+	512MB	RedHat Linux 9.0 (Kernel 2.4.27)
Linux7	AMD Athlon(TM) XP 2600+	2GB	RedHat Linux 9.0 (Kernel 2.4.27)
Linux8	AMD Athlon(TM) XP 2600+	2GB	RedHat Linux 9.0 (Kernel 2.4.27)
Linux9	AMD Athlon(TM) XP 2600+	2GB	RedHat Linux 9.0 (Kernel 2.4.27)
Linux10	AMD Athlon(TM) XP 2700+	2GB	RedHat Linux 9.0 (Kernel 2.4.27)
Linux11	AMD Athlon(TM) XP 2600+	2GB	RedHat Linux 9.0 (Kernel 2.4.27)
Linux12	AMD Athlon(TM) XP 2600+	1.5GB	RedHat Linux 9.0 (Kernel 2.4.27)
Linux16	AMD Athlon(TM) XP 2700+	2GB	RedHat Linux 9.0 (Kernel 2.4.27)
Linux17	AMD Athlon(TM) XP 2700+	2GB	RedHat Linux 9.0 (Kernel 2.4.27)
Linux18	AMD Athlon(TM) XP 2700+	1GB	RedHat Linux 9.0 (Kernel 2.4.27)
Ccbsd1	AMD Athlon(TM) XP 1700+	512MB	FreeBSD 4.11-STABLE

5.2.1搜尋相關

搜尋成功率

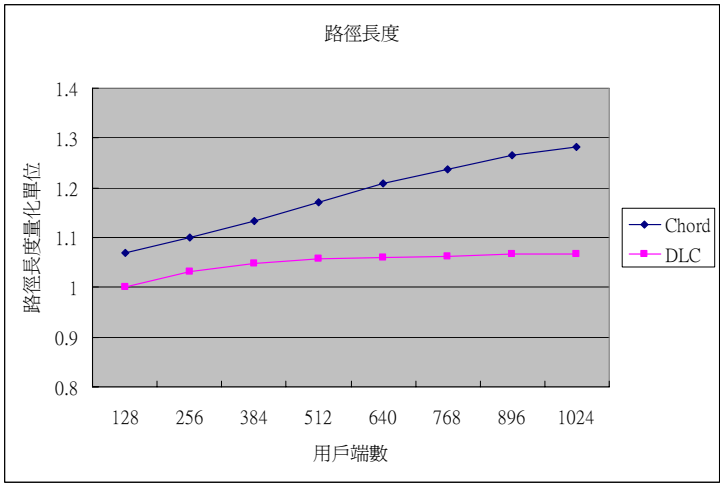
圖表 50 表示 DLC、Chord 的搜尋成功率。圖中 Y 軸表示搜尋成功率並以百分比爲單位。每一點表示該種邏輯架構在某一平均用戶數的情況下，整體的平均搜尋成功率。一般來說，DLC 的效能要比 Chord 來得高，即使在較差的情況下，DLC 的表現至少也能與 Chord 相當。值得注意的是，DLC 在 512 個用戶端之後 DLC 的優勢趨於明顯，應是 DLC 的主動地域代理機制奏效，使得整體效能不因用戶端離開而造成很大的影響。



圖表 50 搜尋成功率

搜尋路徑長度

圖表 51 表示搜尋路徑長度。圖中 Y 軸表示路徑長度量化單位，以 DLC 中 128 用戶端的搜尋路徑長度爲 1，每一點表示該種邏輯架構在某一平均用戶數的情況下，整體的平均搜尋路徑長度。很明顯地，Chord 的搜尋路徑長度因用戶端增多而明顯上昇，反觀 DLC 成長相當緩慢，應該是主動地域代理與雙向搜尋帶來的益處。

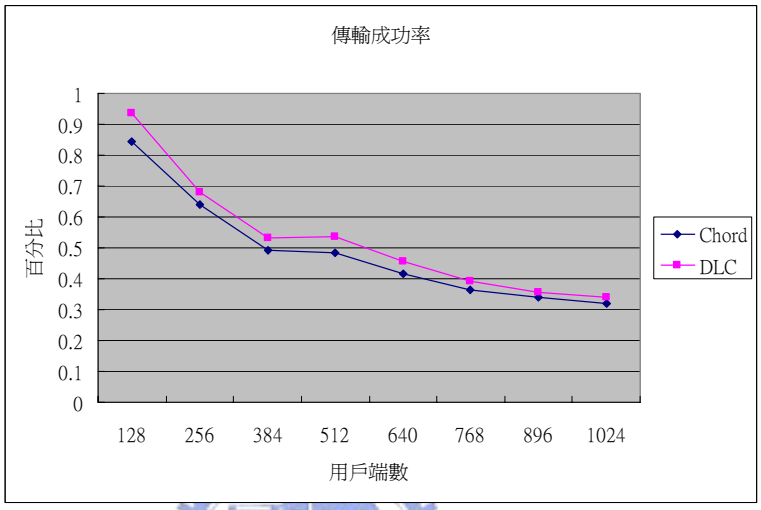


圖表 51 搜尋路徑長度

5.2.2傳輸相關

傳輸成功率

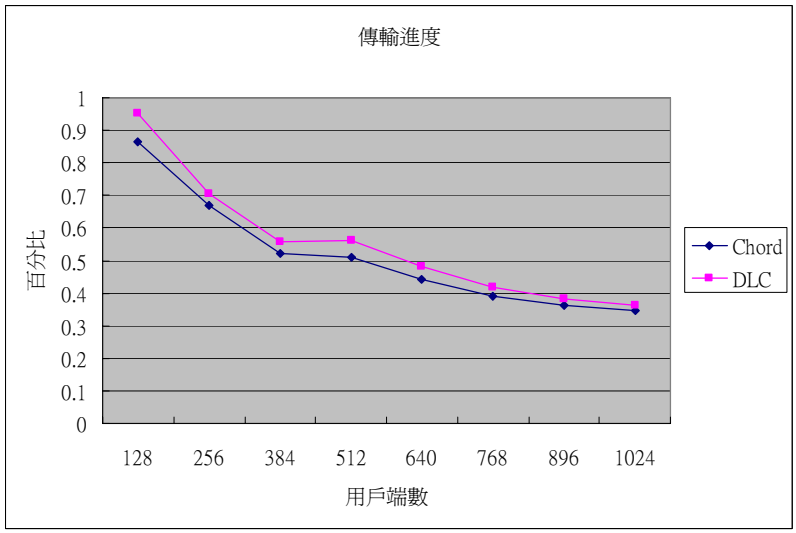
圖表 52 代表傳輸成功率，圖中 Y 軸表示傳輸成功率並以百分比爲單位。每一點表示該種邏輯架構在某一平均用戶數的情況下，整體的平均傳輸成功率。整體來說 DLC 的表現皆優於 Chord，應是 DLC 有主動地域代理帶來的益處。



圖表 52 傳輸成功率

傳輸進度

圖表 53 表示傳輸進度，圖中 Y 軸表示傳輸進度並以百分比爲單位。每一點表示該種邏輯架構在某一平均用戶數的情況下，整體的平均傳輸進度。整體來說 DLC 的表現都優於 Chord。應可歸功於 DLC 有主動地域代理與 BT 追蹤者傳輸，增加檔案來源的存活與傳輸速度。

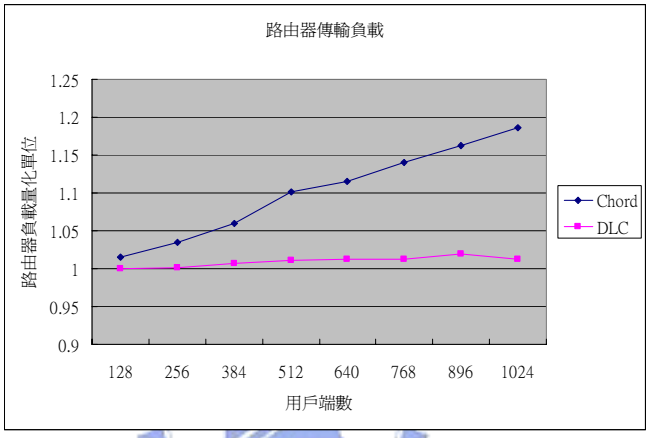


圖表 53 傳輸百分比

5.2.3 負載相關

路由器傳輸負載

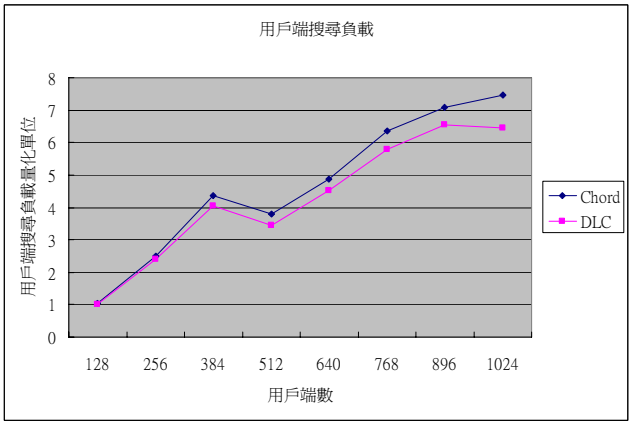
圖表 54 中表示路由器的傳輸負載，圖中 Y 軸表示路由器傳輸負載量化單位，以 DLC 中 128 用戶端的路由器傳輸負載爲 1，每一點表示該種邏輯架構在某一平均用戶數的情況下，整體的平均路由器傳輸負載。而圖中可明顯看出隨著用戶端的增加，兩種網路對路由器的負擔越高，然而 DLC 增加緩慢，Chord 增加明顯。同時亦顯示出，Chord 對整個網路的使用率較高，應爲主動地域代理帶來的好處。



圖表 54 路由器傳輸負載

用戶端搜尋負載

圖表 55 表示用戶端搜尋負載，圖中 Y 軸表示用戶端搜尋負載量化單位，是以 DLC 中 128 用戶端的用戶端搜尋負載爲 1，每一點表示該種邏輯架構在某一平均用戶數的情況下，整體的平均路由器傳輸負載。圖表 55 表示 DLC 與 Chord 用戶端的搜尋負載，基本上 DLC 的負載皆低於 Chord，應爲主動地域代理帶來的好處。



圖表 55 用戶端搜尋負載

第六章結論

6.1 討論

下面我們將 DLC 與四個在第三章討論過的點對點傳輸架構做比較，表格 7 爲歸納結果。N 爲最大用戶端總數。

	Our approach	Chord	CAN	Pastry	Kademlia
參數	Num of Area g	None	Dimension d	Base b	Constant k Base b
搜尋路徑長度	O(log(N/g))	O(logN)	$O(dN^{1/d})$	$O(\log_2{}^bN)$	$O(\log_2{}^bN)$
路由表大小	2*(log(N/g) - 1) + (g-1)	logN	2d	$(2^b-1)\log_2{}^b N$	$k*(2^b-1)\log_2{}^b N$

表格 7 綜合比較

相對於 CAN，DLC 雖然在更新成本與路由表上佔劣勢，但是在搜尋路徑長度上，遠比 CAN 佳。而基本上 DLC 改自 Chord，所以理論分析上，DLC 皆優於 Chord，唯獨路由表的維護上需要耗費較大的成本。如果純看理論數據 Pastry 與 Kademlia 在部分情況下會優於 DLC。但事實上 Pastry 架構和搜尋有關的除了路由表之外，還有 Leaf set、Neighborhood set 兩張子表，因此 Pastry 要維護的相關用戶端總數並非只是表格中的數目。

而Kademlia必須額外維護整個樹狀架構，其維護的成本並不能光以路由表的大小決定。搜尋路徑長度表中寫的$\log_2{}^bN$是指一般狀況下的邏輯樹高，如果邏輯樹產生歪斜，其搜尋路徑長度會快速上升。另外，爲了避免部分子樹的遺失，Kademlia對於一子樹採取重複多記k個用戶端，但是k越大將會使得路由表大小成倍數上升。

總體來說，搜尋路徑長度與路由表大小往往是不能兼顧的。本論文提出的 DLC 是確實可行的，而且能以簡單的通訊協定維護整個架構，在搜尋上不輸給一般常見的點對點傳輸架構，在其他傳輸方面，則可降低用戶端與實體網路的負載，其結果已達到本論文預先設定的目標。

6.2 結論

本篇論文提出的 DLC 主要的精神就是將原先 Chord 加上雙向搜尋、主動地域代理、BT 追蹤者傳輸等三項技術，藉由增加少量的負擔，增進點對點網路的整體效能。經過第五章的模擬之後，我們列出 DLC 可達成的幾個項目：

- 利用雙向搜尋提供兩個方向的搜尋路徑，可以避免遺失一個鄰居而無法對某區域做搜尋，因此可降低搜尋失敗。同時 DLC 的路由表大小相較效能接近的其他點對點網路架構，DLC 顯然減少許多。另外雙向搜尋解決 Kademlia 所提出等距離等搜尋長度的問題，而且不需要額外的通訊協定輔助，除了減低整體通訊協定的複雜度之外，更可減短搜尋路徑，間接亦可降低用戶端的搜尋負載。

- 利用主動地域代理提供邏輯網路搜尋與傳輸的快取機制，可直接降低搜尋與傳輸的負載，間接加速搜尋、傳輸並降低用戶端與實體網路的負載。而地域的功能則可實際加快傳輸速度。另外主動地域代理實質上提供多重代理，所以同樣的一份資料將有八份重複，增加了資料存活率。

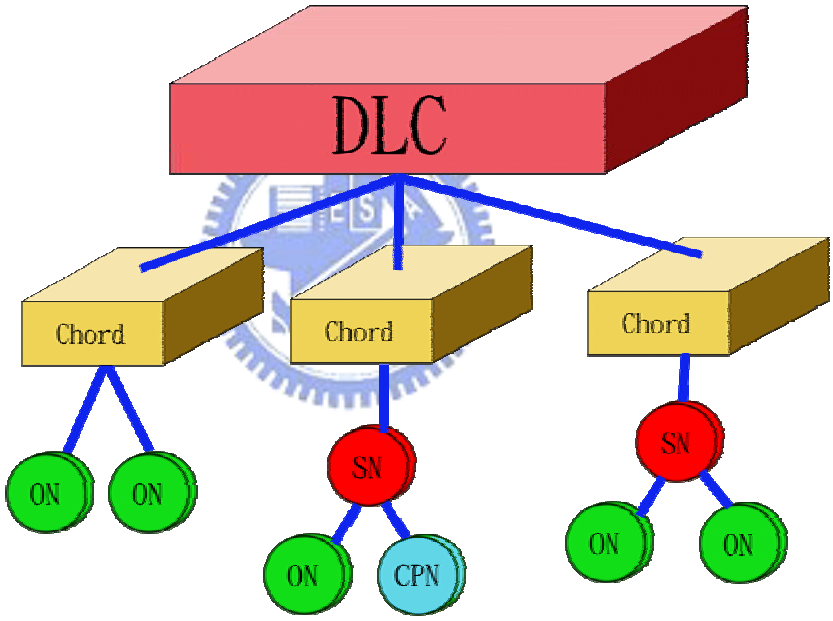
- 利用 BT 追蹤者傳輸機制爲要求傳輸者提供多個來源可加速傳輸、維護分享資源存活，對於實體網路的負載亦能降低。

- 結合主動地域代理與利用 BT 追蹤者傳輸機制，除了有上述優點之外，兩者結合後可以省去原先檔案分享者尋求穩定追蹤者的問題。另外也解決了原先 BT 傳輸中，只靠單一追蹤者來輔助傳輸，造成該追蹤者負載增大的問題。

6.3 未來工作

在實際網路中，用戶端的網路能力差異可能相當大，因此 KaZaA 的 SN/ON 概念可以加入 DLC 的架構，若以 SN 爲 DLC 的一個用戶端，不僅可以避免網路能力差的用戶端影響邏輯網路的表現，更可減少用戶端加入、離開 DLC 的網路成本，增加整體的穩定性。不過 SN/ON 架構中 SN 的選取是必須解決的前提。

另外 DLC 的雙層架構，可做進一步的延伸類似 NAT 的概念，如圖表 56 中階層式架構，除了可以擴大邏輯空間，較低層則可採用簡單邏輯網路來降低用戶端的負載。此外亦可建立特殊的用戶端 CPN (Cross Platform Node)，用以銜接不同的點對點邏輯網路做到真正的廣域搜尋。



圖表 56 階層式點對點傳輸架構

參考文獻

[1] CAN : S. Ratnasamy, P. Francis, M. Handley, R. Karp, amd S. Schenker, “A Scalable Content-Addressable Network,” In Proceedings of SIGCOMM 2001, ACM. pp. 168-175.

[2] RI：A. Crespo and H. Garcia-Molina. Routing Indices For Peerto -Peer Systems. In Proceedings of the 22nd International Conference on Distributed Systems, pages 23--32, Vienna, Austria, 2002.

[3] Chord : Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan, Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, ACM SIGCOMM 2001, San Deigo, CA, August 2001, pp. 149-160

[4] Pastry : A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems". IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, pages 329-350, November, 2001

[5] Kademlia：P. Druschel, F. Kaashoek, and A. Rowstron (Eds.): IPTPS 2002, LNCS 2429, pp. 53–65, 2002. Springer-Verlag Berlin Heidelberg 2002

[6] Narada : Yang-Hua chu, Sanjay G. Rao, and Hui Zhang, “A case for end system multicast,” in ACM SIFMETRICS, 2000, pp. 1-12

[7] Understanding KaZaA：J. Liang, R. Kumar and K.W. Ross, "Understanding KaZaA," submitted, 2004

[8] Napster. [Online]. Available : http://www.napster.com/

[9] Gnutella. [Online]. Available: http://www.gnutella.com/

[10] Emule [Online]. Available: http://www.emule-project.net/

[11] Edonkey. [Online]. Available:http://www.edonkey2000.com/

[12] WinMax. [Online]. Available: http://www.agry.purdue.edu/max/

[13] Kuro. [Online]. Available: www.kuro.com.tw/

[14] Ezpeer. [Online]. Available: http://www.ezpeer.com/index.html

[15] Skype. [Online]. Available: http://www.skype.com/

[16] MSN. [Online]. Available: http://www.skype.com/

[17] ICQ. [Online]. Available: http://www.icq.com/

[18] FastTrack [Online]. Available:http://www.slyck.com/ft.php?page=1

[19] KaZaA [Online]. Available : http://www.kazaa.com/us/index.htm

[20] Morpheus [Online]. Available : http://www.morpheussoftware.net/

[21] Bittorrent [Online]. Available : http://www.bittorrent.com/

[22] Yahoo Messenger [Online]. Available : http://www.yahoo.com/

[23] KaZaA-Lite [Online]. Available:http://c2p.6x.to/

[24] coolstreaming [Online]. Available : http://www.coolstreaming.org/

[25] Gnutella PURE PEER TO PEER PROTOCOL
http://www.ece.rutgers.edu/~parashar/Classes/01-02/ece579/slides/gnutella.pdf

[26] Baytsp http://www.baytsp.com/

[27] SkypeOut and SkypeIn Gateways
http://463west.blogspot.com/2005/05/skypeout-and-skypein-gateways.html

[28] Pastry Routing Table
http://www.scs.cs.nyu.edu/V22.0480-005/notes/l24.pdf

[29] P2P 進入第三代　功能更悍 [Online]. Available:
http://taiwan.cnet.com/news/ce/0,2000062982,20093088,00.htm

[30] U.S. Court of Appeals for the Ninth Circuit
http://www.ce9.uscourts.gov/web/newopinions.nsf/0/
c4f204f69c2538f6882569f100616b06?OpenDocument