Crown ：分散式格網服務搜尋協定
（Crown：a Stable Distributed Grid Service Lookup Protocol）

李易寰　　蔡昆樺　　王宗一
智慧型網路應用實驗室
國立成功大學工程科學系
台南市大學路一號
TEL: (06)2757575-63338
E-mail:n9691112@ccmail.ncku.edu.tw
E-mail:oliver@lina.es.ncku.edu.tw
E-mail:wti535@mail.ncku.edu.tw

摘要

格網計算是目前相當熱門的研究領域，而如何在格網環境下快速的搜尋找到使用者所需的運算資源已成為一個很重要的議題。本文提出一個基於分散式並且以群組群環狀(Group-Ring)拓樸為基礎的網路資源搜尋協定、稱之 Crown。透過 Crown，使用者可以在格網環境中快速的搜尋到想要且合適的網路資源來使用，而且也可以降低個別資源在加入與離開的網路時的負擔，因此 Crown 具有優良的擴充性(scalability)，另外在協定內也設計複本與回復機制加強整個網路的可用性(availability)及強固性(robustness)。

關鍵詞：格網服務，服務搜尋協定，分散式網路

一、簡介

近幾年，**格網服務（Grid Service）[2]**相當受到重視，在網際網路技術及速度日益精進下，如何讓網路中的閒置資源被充分利用，已成為最熱門的研究領域之一。而要完成上述的共享應用有一個很重要的研究課題—就是如何在網路上方便快速地找到使用者想要而且適用的資源。也就是說要讓使用者在 Grid 網路中，使用各式各樣的服務或資源時，必須要有一套快速、穩固、並且可幸賴的資源搜尋機制作為基礎建設的一環。

在格網計算目前的發展，基本上多數研究還是屬於集中式的搜尋方式，但是在其他分散式網路的應用，例如點對點（P2P）網路應用，目前已經有很多研究提出許多不同的拓樸與協定、應用於 P2P 網路資源搜尋。仔細考慮格網的整體網路環境，可看出其基本上也是類似於分散式計算網路環境，因此本文提出一適用於格網運算的資源搜尋協定-- Crown。Crown 是一個基於 Chord Protocol **[15]** 所發展出的機制協定，將其原先的網路拓樸由環狀(Ring)擴充成群組環狀(Group-Ring)，另外再加入複本(Replica)及錯誤回復機制(Error Recovering)來強化資源搜尋的速度、可靠、及可用性。並且透過分散式雜湊表(Distributed Hashing Table -DHT)的機制，Crown 將搜尋網路資源的工作分散給網路的某些節點（Peers）、並讓它們一起合作，因此不會像原先格網運算所使用的其他方法、造成某些節點工作量大增。另外 Crown 的群組設計也可以讓使用者、資源等，在加入或離開格網時，整個格網計算環境還是可以很平順的運作，不會造成額外的負擔。

二、相關研究

目前在分散式網路上所提出的網路資源搜尋架構大致可系分成三種類型，茲分別介紹如下。

(一) 集中式類型（Centralized）

集中式類型是指每一個節點要取得網路上的資源時，必須先像一個特定的伺服器(server)查詢，當查詢訊息回應回來後，則節點可以直接和搜尋到的節點直接連線取得資源而不需再透過伺服器的斡旋轉達。

Napster [9]算是使用此方式最出名的 P2P 系統，透過 Napster 伺服器的查詢服務，使用者可以查詢到他想要的 MP3 音樂所在位置，然後再直接連到此節點直接存取。另外像是 Web Service [5]中的 UDDI [6]機制也是使用 SOAP 協定向特定的 UDDI Node 來查詢服務內容。使用集中式類型的優點在於使用者可以容易快速的搜尋到大量相似的資源，但是缺點

則是提供查詢服務的伺服器會成為此系統的瓶頸，如果伺服器發生故障或是網路頻寬不足，都會大大降低整個系統的可用性(availability)與強固性(robustness)。

(二) 廣播式類型（Flooding or Gnutella-like）

顧名思義，廣播式類型便是透過廣播的方式由節點向其鄰近相連的節點發出搜尋的訊息，如果其鄰近的節點有此資源則會回傳訊息給發出要求的節點，否則就將訊息在發送給其鄰近的節點，以此類推。

Gnutella [16] 是最具代表性的系統，透過廣播的方式，要求資源的訊息會被散佈到有相連的節點上，每一個節點收到要求訊息會立刻檢查本身是否擁有要求的資源，如果存在的話則回傳訊息回去，否則就將要求訊息再散佈出去，一些改良 Gnutella 的系統則是增加所謂歷史紀錄，例如接收到要求資源訊息的節點如果之前也曾要求過類似的資源，那他便能直接回傳這些節點的資訊，而不需在散佈要求訊息出去，而每一個要求訊息都有一個存活期(TTL)的紀錄，一旦訊息流動的次數超過存活期則此要求訊息將被捨棄。

這一類的系統最大的好處是作到完全的分散式，每一個節點只需管理自己的資源，但是由於是採取廣播的方式，雖然每一個要求訊息會被存活期限制住，但在廣域網路上還是容易產生訊息氾濫的現象增加整個網路的負擔，其次由於存活期的限制可能造成每次只能搜尋到部分資源，或甚至找不到所需資源。

(三) 分散式雜湊表類型（Distributed Hash Table Based，DHT）

目前許多分散式資源搜尋的研究多採用此種方法，使用雜湊表與生俱來的好處，就是可以達到平衡負載(Load Balancing) [7] 的效果，本論文所提出的 Crown 也是屬於此種類型，透過分散式雜湊表的方式，每一個節點及提供的資源都會經由雜湊函數(Hash Function)被轉換成一串的數值（索引值），這些索引值 s 會被放置於某一些特定的節點 s 上，當使用者想要搜尋一資源時，所輸入的資源資料會被轉換成索引值，透過分散式雜湊表的對應，便能很快的查詢到此資源的所在位置。

分散式雜湊表機制較早見於 Plaxton [10] 等提出之搜尋機制，之後的 Chord、CAN [12]、Tapestry [17]與 Pastry [14] 都是採用分散式雜湊表相當有名的幾個協定，利用此方式的架構本質上都屬於分散式架構，在搜尋資源時既不像集中式的類型需考量伺服器的問題，與廣播式比較又不容易造成網路訊息氾濫的問題，因此在搜尋資源的成本上是屬於較好的方式，但是使用分散式雜湊表機制其缺點在於當鄰近節點有變動時，需要較多的通訊來處理維護鄰近節點的資訊。在格網服務的環境中節點的變動是非常頻繁的，因此如何減低節點變動時所造成的網路負擔便是一重要的課題。CAN 採用的邏輯拓樸可以避免這個問題，但 CAN 相對存在有繞送路徑過長的問題，Pastry 也嘗試解決這個問題，但其節點加入的過程較為繁複。Crown 主要的貢獻之一就是利用分散式雜湊表的架構，維持網路資源搜尋上的速度，同時也提出克服節點變動頻繁時所造成的問題的方法。

三、Crown 邏輯覆蓋網路（Crown Overlay Network）

在這一章節，將首先描述 Crown 使用的邏輯覆蓋網路(Crown Overlay Network)，並介紹其中節點的類型以及如何決定節點在邏輯覆蓋網路中的角色類型。接著介紹所使用的分散式雜湊演算法如何產生節點唯一名稱與格網服務資源的索引值（索引值），最後說明在格網中服務資源索引值的散佈規則與如何搜尋這些索引值來獲取格網服務。

(一) Crown 拓樸架構（Crown Topology）

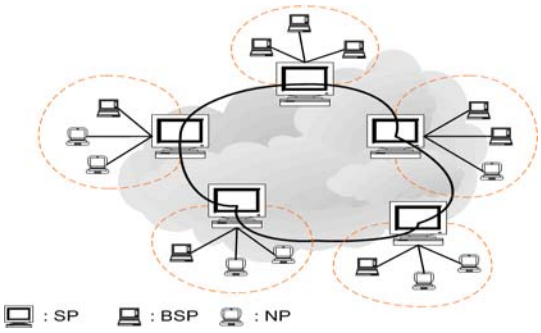


圖 3.1：Crown 拓樸架構

如圖 3.1 所示，Crown 使用的拓樸基本上是由多個群組所構成的環狀的架構（Group Ring-liked），搜尋服務的訊息只會在環狀上跳躍繞送（Jump & Forwarding），而群組架構是用來降低節點加入或離開時的網路負擔，在 Crown 拓樸中節點被分成三種類型，而同一時間一個節點只會屬於群組主節點(Super Peer-SP)或是群組備份主節點(Backup Super Peer-BSP)其中一種類型，以下是三種類型的介紹：

- **群組主節點（Super Peer，SP）**：群組主節點負責維持拓樸中環狀架構的連結，搜尋服務的訊息會在群組主節點中跳躍

繞送，另外群組主節點也會維護一些和他鄰近的群組主節點資訊來維持 Crown 的運作，並且在每一個群組中同一時間只允許一個群組主節點的存在。

- **群組備份主節點（Backup Super Peer，BSP）：**群組備份主節點是存在於拓樸中的群組架構部份，群組備份主節點主要的作用是為了預防群組主節點錯誤或離開時能快速接手原本群組主節點的工作，保持 Crown 運作的順暢，在群組架構中群組備份主節點可能不只一個，在稍後的章節會詳細描述如何決定群組備份主節點的數目。
- **一般節點（Normal Peer，NP）：**一般節點則是群組架構中一般的成員，基本上這些節點只需維護管理本身的資源。

在格網服務環境中，節點的加入與離開是非常頻繁的（考量目前行動設備的快速增加，節點更動頻繁是可以想像的），因此若是僅單純使用環狀拓樸，則節點的加入與離開上將會造成環狀拓樸的斷裂，整個環狀網路將一直處於更動的狀態下，所以本文將環狀拓樸擴充成為群組環狀拓樸，搜尋服務的訊息依然是在環狀上繞送，仍然可以保證擁有一定程度的服務搜尋速度，此外利用群組架構，可以降低環狀的大小、也就是訊息繞送的次數外。節點加入或離開時其需要的資料變動可以限制在群組架構中而減少環狀的變動，讓整個網路中節點的加入與離開變得更有效率，進而增加整個 Crown 在搜尋格網服務的可獲得性及強固性。

(二) 分散式雜湊表格(Distributed Hashing Table)

在前面的章節簡單的介紹了分散式雜湊表的概念與格網服務索引值產生的目的，使用分散式雜湊表好處在於可將所需資源名稱或其他關鍵字透過雜湊函數轉換成獨一無二的索引值，經由索引值的比對來達到有效率的繞送及搜尋，並且達到平衡負載的效果。因此雜湊函數的選擇就相當關鍵，經參考[15] [17]及比較各種雜湊函數的性質後，發現 SHA-1 [13]為合適的演算法。SHA-1 可將不同的輸入值轉成獨一無二(Unique)的輸出值，並且其輸出值的分佈(distribution)相當平均。因此 SHA-1 相當適合用在分散式雜湊表格的機制上。

(三) 群組名稱（Group ID）與節點名稱（Peer ID）的產生

Crown 協定中的節點都有一個節點名稱及一個群組名稱，每一個節點是直接以本身的實際 IP 位址當作節點名稱，而群組名稱是透過 SHA-1 得到。以目前 IPv4 IP 位址的格式來說，藉由 SHA-1 輸入位址前 m bit 來產生群組名稱，因此在每一個群組中將最多會有 $2^{(32-m)}$ 的節點數目，目前本文是採用位址前 24 bit（即 m=24）來產生群組名稱，如圖 3.2 所示：

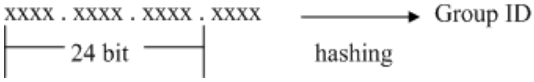


圖 3.2：群組名稱產生方式

原本環狀拓樸節點數量過多時會影響到整個環狀上訊息傳輸與節點運作的效率，因此使用群組架構來容納局部區域的節點個數，如上圖 3.2 所示，雜湊位址前 24 bit，如此即可減少環上節點的數量，另外也可達到鄰近的位址節點會落於同一群組的效果。 以圖 3.2 的例子來說，一個群組的大小剛好是一個 Class C 子網路的大小，群組中的節點都在同一個 LAN 之內，因此群組內，節點間的通訊相當快速。在考量 m 值的大小時，不一定需要是 24 bit，但是本文建議最好是介於 16 bit 至 24 bit 之間，在這個範圍內的節點很大的機率會處於鄰近的區域，因此可以保證節點之間的通訊連線品質可以維持在一定程度上，這對於稍後下一章節所提到的複本及回復機制將會有相當的幫助。

(四) 群組主節點評估推舉機制（SP Evaluation Mechanism，SPEM）

在 Crown 協定中，群組必須推選一個群組主節點來維持對 Crown 邏輯覆蓋網路的連結。如果群組中只有一個節點，則它一定會成為群組主節點，但是若群組中有許多節點，則採取一評估機制，來決定群組主節點，其主要是依照以下五個參數來決定成為群組主節點的優先順序：

- **網路的頻寬與使用狀況（Bandwidth and Utility）：**在正常狀況下，頻寬越大或是使用率較低的節點，應該先被優先選為群組主節點。
- **可獲得性（Availability）：**節點連線的時間長短（Up-time）或是其歷史紀錄的評價越高者（History Record）應該優先成為群組主節點，歷史紀錄的評價則包括常發生斷線或錯誤與否，比較穩固的節點應該優先選擇成為群組主節點。
- **計算能力（Computing）：**節點計算能力越強，表示處理訊息的速度越快，也比較不會因為處理大量的訊息而發生錯誤。
- **設備的特性（Device Characteristic）：**由

於現在行動設備（Mobile Devices）的增加，有些節點是屬於此類型的設備，而行動設備的特性便是具高度的移動性，因此其容易在多個網域中移動，並不適合成為群組節點或是群組備份主節點。

- **使用者設定（User Setting）：**此參數比較特別，是由使用者來決定是否有意願成為群組主節點。

上述中第 1、2 項參數可利用 SNMP 網路管理中所使用的 Health Function [4] 來取得參考值，第 3、4 項則可經由存取系統的資訊來取得，若使用的為行動設備則其第四項參數值會較低，除非在群組中的節點數目不足時，否則這些設備不會成為群組主節點或群組備份主節點，第 5 項則是使用者選擇，若使用者不願意其設備成為群組主節點，除非在沒有其他節點可選擇的情況下，否則不會成為群組主節點。

(五)索引值散佈機制（Key Distributed Mechanism）

Crown 是將節點上的服務的內容（例如檔案名稱、服務程式名稱等）透過 SHA-1 產生一個索引值，接著再將此索引值放置在 Crown 邏輯覆蓋網路中一個對應的節點上，資源內容的轉換則須視所應用的目的為何。本文的目的是應用在格網環境中搜尋可利用的格網服務，因此採用像電話簿黃頁的分類方式將服務劃分在許多子類別中，再針對這些子類別及服務程式名稱透過 SHA-1 作轉換。

當一個格網服務被轉換成一索引值後，會依照下面的規則來散佈此索引值：

(1) 首先搜尋 Crown 邏輯覆蓋網路中是否有群組主節點的群組名稱的值等於此索引值，如果有則將索引值放置於此群組主節點上。

(2) 若沒有等於索引值的主節點，則搜尋目前 Crown 邏輯覆蓋網路中群組主節點群組名稱比索引值大，而距離（|群組主節點群組名稱–索引值|）最小者。

透過這簡單的散佈機制，索引值會很容易的散佈在 Crown 邏輯覆蓋網路，並且根據 SHA-1，可以保證索引值的均勻分布。

(六)索引值搜尋機制（Key Lookup Mechanism）

在 Crown 邏輯覆蓋網路中，搜尋訊息透過群組主節點跳躍繞送，並不會被繞送至群組中的節點，而繞送路徑則是向訊息繞送表（Message Forwarding Table）查閱。

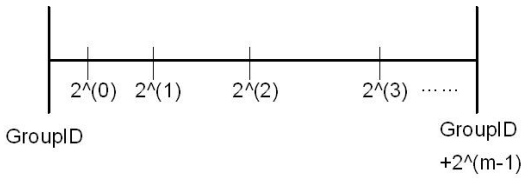


圖 3.3 訊息繞送表紀錄的區域範圍

訊息繞送表會紀錄至多 m（$=logN$，N 是群組的數目）筆記錄，每一筆資料記錄代表一個區段範圍。如圖 3.3 所示，範圍的大小是以 2 的次方成長，每一塊區段負責的群組主節點為群組名稱$+2^{(m-1)}$（假設群組主節點為 G32，則其第一個紀錄的區段要參考的群主主節點等於 $32+2^{(1-1)}=33$，因此要參考 G33 的群組主節點），但是如圖 3.4 所示，G33 此群組目前並不存在於 Crown 邏輯覆蓋網路中，因此必須選擇第一個比 G33 名稱還大的群組的群組主節點來負責（在圖 3.4 中第一個比 G33 大的是 G40，因此 G32 的訊息繞送表中，其第一筆紀錄為 G40），這樣選擇的原因是要符合前一節中描述索引值放置的規則，如此訊息才會被正確繞送。假設要查詢的索引值若位於第一塊範圍，則群組主節點會將訊息轉傳給負責第一塊範圍的群組主節點，以此類推，在訊息繞送表上每一筆紀錄有三個欄位：

- **群組參照（Group Reference）：**每個區段負責的群組主節點，所要求索引值落於此區段的，便會被繞送至此主節點上。
- **群組主節點參照（SP Reference）**：用來紀錄區段負責群組主節點的實際位址。
- **群組備份主節點參照（BSP Reference）：**群組中會有群組備份主節點支援群組主節點，當訊息被繞送失敗時，則會使用群組備份主節點位址來嘗試繞送訊息。

在說明訊息繞送表的意義後，接下來以圖 3.4 來描述在服務索引值的搜尋過程。

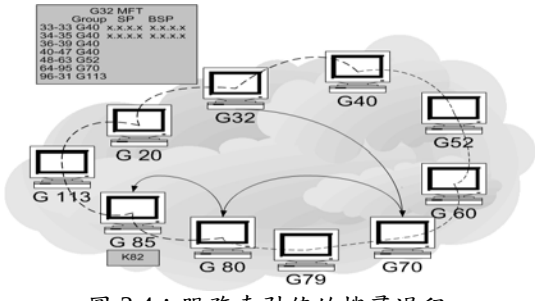


圖 3.4：服務索引值的搜尋過程

(1) 首先節點根據訊息繞送表的資訊，發出資源要求的訊息，在圖 3.4 中，假設群組 32（G32）中的某一個節點要搜尋某一服務，此服務經 SHA-1 轉換成索引值 82（K82），則搜尋機制將搜尋 K82。

(2)搜尋機制根據以下的原則來繞送訊息，由訊息繞送表中的紀錄，第一個小於或等於的群組將被選擇成為下一個訊息繞送的地點，如圖 3.4 左上所示，經檢查發現 G70 是第一個小於 K82 的群組主節點，因此選擇將訊息繞送至 G70，接著以此類推，在 G70 上也是透過相同的規則進行繞送，最後會在 G85 上找到 K82，因此至多只需經過 O(log N) 次的繞送便回得到結果。

四、加入/離開與錯誤回復機制（Join/Leave and Recovery Mechanism）

本章節中描述節點加入與離開 Crown 邏輯覆蓋網路的詳細步驟。並討論 Crown 的錯誤回復機制來處理可能發生的錯誤情形。

在 Crown 邏輯覆蓋網路中群組主節點的主要任務為負責維持 Crown 邏輯覆蓋網路中環狀的連結，並管理與儲存群組中所存放的服務索引值，群組主節點會將群組中所存放的索引值存放成一個列表，稱之為索引值表（Key List）。在錯誤回復機制中，為了推舉新的群組主節點以及選取群組備份主節點，群組主節點須紀錄群組中每個節點經過群組主節點評估機制所得到結果，並加以排序存放，稱之為推舉順序表（Top List）。

(一) 節點加入 Crown 邏輯覆蓋網路

本文假設一節點能夠取得目前 Crown 邏輯覆蓋網路上某一個節點的資訊，例如透過特定的伺服器或是經由節點之前的歷史紀錄或是其他的方法，這個節點一般被稱為起始化節點（Bootstrap node）。以下則是節點加入的詳細步驟：

(1) 節點經 SHA-1 得到對應的群組名稱。

(2) 節點發出要求加入的訊息給起始化節點，查詢要加入的群組主節點實際位址。

(3) 根據節點加入時查詢的結果會有兩種不同的情況，第一種狀況是在此 Crown 邏輯覆蓋網路上目前並沒有此節點所屬的群組，第二種狀況則是節點所屬的群組已經存在，第一種情況則繼續第(4)步驟，否則跳至第(5)步驟。

(4) 節點欲加入的群組不存在，則此加入節點必須成為其群組的群組主節點，因此執行下面的初始群組主節點步驟：

1. 節點透過起始化節點建立訊息繞送表，節點會請求起始化節點查詢建立訊息繞送表所需要的資訊。例如在圖 4.1 中，假設欲加入的節點要成為 G36 群組主節點，而起始化節點為 G99 中的任一個節點，首先 G36 群組主節點送出要求訊息給位於 G99 中的起始化節點去查詢 G37、G38、G40、G44、G52、G68、G100、G164 的群組主節點資訊，用以建立 G36 群組主節點訊息繞送表。

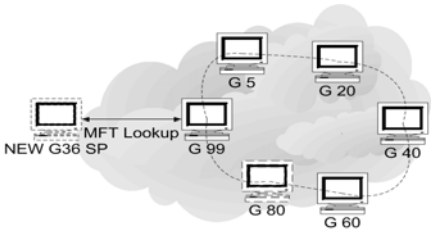


圖 4.1 新加入的群組主節點透過起始化節點建立訊息繞送表

2. 如圖 4.2，G36 群組主節點完成訊息繞送表建立後，會要求存在於其訊息繞送表的群組主節點更新它們本身的訊息繞送表的資訊，同樣的其訊息繞送表中的群組主節點再遞迴的要求更新。

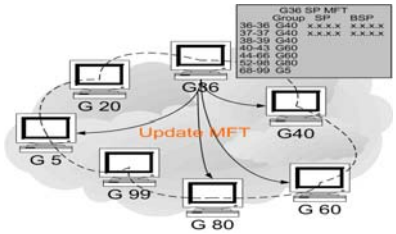


圖 4.2 訊息繞送表的更新

3. G36 群組主節點再與相鄰群組主節點通訊，將應該存放於 G36 群組主節點的索引值傳送過來，如圖 4.3 所示 K30 將從原本 G40 群組主節點位置改放於 G36 群組主節點位置。

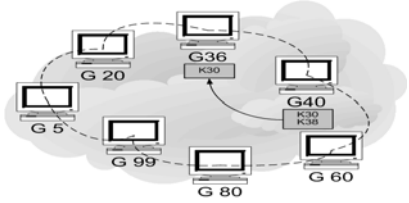


圖 4.3 格網服務索引值的轉移

4. G36 群組主節點初始其推舉順序表，並依據群組主節點評估機制得到一個評估的分數，之後當有新的節點加入此群組時都會依據評估機制產生一分數紀錄於推舉順序表，作為推舉新群組主節點依據。

5. G36 群組主節點將要分享的資源，經 SHA-1 轉換成相對應的索引值，並依照訊息繞送表，將這些索引值發到所應該存放的群組位置。

(5) 第二種情況，新節點要加入的群組已存在，其加入步驟如下：

1. 新節點送出加入的訊息（包括節點本身被評估的分數值）給其所屬群組的群組主節點，群組主節點會回傳目前的訊息繞送表給要加入的節點。

2. 群組主節點更新其推舉順序表的紀錄。

3. 新節點將要分享的資源，經 SHA-1 轉換成相對應的索引值，並依照訊息繞送表，將這些索引值發到所應該存放的群組位置。

(二) 節點離開 Crown 邏輯覆蓋網路

如果節點是群組備份主節點或一般節點，它僅需通知其群組中的群組主節點後，便可離開。但是若離開的節點是此群組的群組主節點，則便需透過回復機制來維護群組的相關資訊，下面就各種群組主節點離開時可能發生的情況加以討論：

(1) 假若群組主節點要離開時，若群組中存在群組備份主節點，則群組主節點會將其管理的索引值表與推舉順序列表遷移到評估值較高的群組備份主節點上，然後將群組主節點的任務轉移給此群組備份主節點。

(2) 假若群組主節點要離開時，群組中已經沒有其他節點，此時群組主節點會將索引值遷移到拓樸中下一個群組的群組主節點，通知此群組主節點更新其資訊。

(3) 若是群組主節點是發生錯誤而非正常離開，則繞送訊息到此群組時便會發生 timeout 的情形，則發送此訊息的節點可以利用其訊息繞送表上群組備份主節點的記錄聯繫，若聯繫成功，則訊息便可直接延續，否則便須等到群組中新的群組主節點產生。

(4) 極端的情況下群組主節點與群組備份主節點同時發生錯誤，則有可能是整個群組發生發生網路分割狀態（Network Partition），此時只有先將此群組從 Crown 邏輯覆蓋網路上移除。當發生這種情形時，整個位於群組中的索引值表就會全部遺失，為了解決這種情形，在後章節會說明使用複本機制來解決。

(三) 群組內回復機制

群組主節點可依對資源可用性的需求，來決定群組備份主節點的數量，當需要越高的可用性時，則可選取越多的群組備份主節點。參考其它論文的作法 [11]，並考量本論文的設計。本文提出一公式來計算群組備份主節點數目建議值：

$$(1)\ (1-(1-p)^k) \geq AT \ ,\ N-1 \geq k$$

其中 k 為群組備份主節點個數，p 為節點正常運作的機率，AT 為可用性最低限度（Availability Threshold），**N** 為群組中節點數量。假定節點正常運作的機率（p）為 70%，則節點發生錯誤的機率即為$(1 - p) = 30\%$，可用性最低限度的要求為 99%，由公式(1)所得到的群組備份主節點最小建議數目為 4 個。當然選取越多群組備份主節點，Crown 邏輯覆蓋網路中資源的可用性會越高，但相對越多群組備份主節點則所需花費的通訊成本越高，因此可針對應用的場合，調整可用性的最低限度，來得到適當的群組備份主節點數目。

在選定群組備份主節點數目後，群組主節點會在每個群組備份主節點上儲存一份其負責索引值表的完整複本，其好處為假若群組主節點發生錯誤時，任一個群組備份主節點都有能力馬上接手群組主節點的工作，使搜尋資源的服務不至於中斷，圖 4.4 顯示一個 k=4 群組主節點與群組備份主節點之間的連結關係，除了群組主節點是必須和每一個群組備份主節點相連外，每一個群組備份主節點之間也是都必須互連，各點間的關係屬於完整連結圖（Complete Graph）。

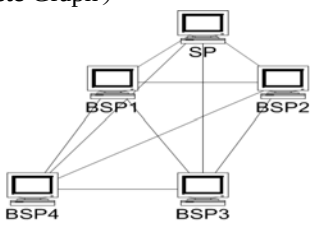


圖 4.4 群組主節點與群組備份主節點關係圖

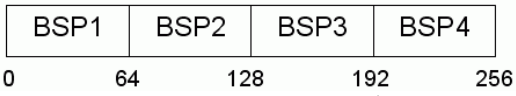
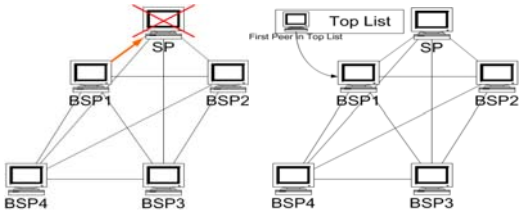


圖 4.5 複本分區段更新策略

群組備份主節點上的複本更新採用分段更新的策略，來減少資料傳遞的成本。如圖 4.4 群組備份主節點有 4 個時（k=4），群組主節點會把其管理的索引值表跟推舉順序列表分成 k 個區段，每一個群組備份主節點只負責更新其中 $1/k$ 的區段部分的資料。如圖 4.5 所示，當其索引值表或推舉順序列表有更新時，

群組主節點就會發出更新訊息給負責該區段的群組備份主節點，通知其更新資料。

當群組主節點離開或錯誤時，群組備份主節點中評估分數值最高者會取代原先的群組主節點，如圖 4.6(a)所示。並由推舉順序列表中選取評估分數值最高者來取代此群組備份主節點的位置，如圖 4.6(b)所示。群組備份主節點取代群組主節點後，在快速的跟其他群組備份主節點更新索引值表跟推舉順序列表其他部分的最新的資料以確保資料的一致性。新的群組主節點產生後，通知訊息繞送表中所紀錄的群組主節點更新其新的群組主節點資訊。



(a) (b)

圖 4.6 群組主節點的回復與遞補機制

(四) 群組間回復機制

在某些應用中，索引值需要被保證有一定程度的可獲得性，在 Crown 邏輯覆蓋網路中當每一個群組只有一個節點時，則整個邏輯覆蓋網路會簡化為一個單純的環狀，在這種情況下，若是其中一個節點錯誤，則存放於此節點上的索引值表就會丟失；或是當發生網路分割(Network Partition)狀態時，則整個群組的索引值表將會無法存取，此時就必須使用群組間回復機制來解決這樣的錯誤。

群組間回復機制的做法是將群組中的索引值表之複本，擺在其他群組的群組主節點上作為備份。一開始索引值在散佈時，除了原本應該放置的位置要放置一份外，還必須放一份在補數運算(2's compoment)後新索引值要放置的地方，並註明其為複本索引值，來避免與一般索引值相衝突。當訊息繞送發生 Timeout 時，則將要查詢的索引值值作補數運算，把計算出來新的索引值取代原本的索引值，然後以新的索引值重新搜尋。由於索引值表複本要放在不同的群組主節點上，網路通訊可能不像在群組中那麼快速，所需的通訊成本較高，因此需考慮實際應用對索引值表可用性的要求是否很嚴苛，決定是否要採用複本備份。

圖 4.8 說明了群組間回復機制運作情況，當 G5 中的節點要搜尋 K37（10011_2），K37 存放的 G40 發生錯誤，搜尋訊息的繞送發生 Timeout，此時將 10011_2 作補數運算得到 1100_2（24_{10}），即可得知 K37 的複本擺放於 K24 的位置，如此就可以順利的在 G36 找到 K37 的複本（K37R）。

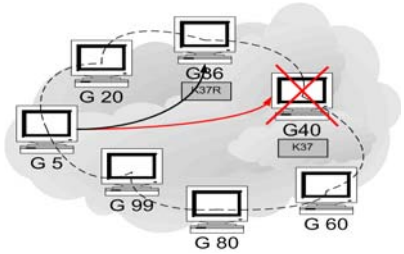


圖 4.8 群組間回復機制

五、效能評估與比較（Performance Evaluation and Comparison）

這節中將比較 Crown 與其他使用分散式雜湊法為基礎的搜尋機制間的差異，透過數學運算的推估，可以看到量化的比較結果。各搜尋機制採用不同的邏輯拓樸，因此不同拓樸間會有不同的參數(Parameter)設定。Crown 中使用群組機制，因此必須設定群組的大小，假設 g。CAN 系統的邏輯拓樸是由 d 維度的座標系統(d-dimensional Cartesian 群組主節點 ace)所組成，所以必須設定所採用的維度(d)。而 Pastry 與 Tapestry 兩系統都必須設定所選用的基底(Base)的值，用 b 來代表。透過各項客觀性質的呈現，可以比較出不同搜尋機制間的差異，下面將定義各項性質：

- **繞送路徑長度（Path Length）**：一次搜尋所經過的節點數，稱為繞送路徑長度。
- **訊息繞送表的大小(MFT Size)**：在搜尋機制中，為了要達到正確且有效率的繞送路徑，所必須維護訊息繞送表的大小，在不同的搜尋機制中有不同的名稱，但其代表的意義是相似的。
- **節點變動所需的通訊成本(Communication Cost)**：當節點加入或離開，需要發出一些訊息來通知其他節點，以維護繞送的正確與效率，訊息量的大小即為所需花費的通訊成本。表一為搜尋機制間性質差異的比較。

表一：DHT 搜尋機制性質比較表

	Crown	Chord	CAN	Pastry	Tapestry
Parameter	Group Size g	none	Dimension d	Base b	Base b
Path Length	$O(\log(N/g))$	$O(\log N)$	$O(dn^{(1/d)})$	$O(\log_{2^b}(N))$	$O(\log_b N)$
MFT Size	$\text{Log}(N/g)$	Log N	2^d	$(2^b-1)\text{Log}_{2^b}(N)$	$b\text{Log}_b(N)$
Join/Leave Communication Cost	$O(\log^2(N/g))$	$O(\log^2 N)$	$O(d)$	$O(\log_{2^b}(N))$	$O(\log_b{}^2 N)$

接著假設初始數據，使用表一的公式，

推估各搜尋機制的表現數據。假定邏輯網路中有 216 個節點 (N=16)。Crown 中群組名稱是雜湊位址前 24 bit 來產生，因此每一個群組裡的節點數目為 2^8(假定在理想狀態，所有群組皆為填滿)。CAN 使用 2-D 座標空間，因此 d=2，Pastry 與 Tapestry 都使用 b=2 為基底。假定節點加入/離開的機率為 0.05/per second，發出通知訊息的大小為 1 k。表二則為搜尋機制表現數據的比較。

表二：DHT 搜尋機制性質比較表

	CROWN	Chord	CAN	Pastry	Tapestry
Parameter	$g = 2^8$	none	d = 2	b = 2	b = 2
Maximum Path Length	8	16	512	8	16
MFT Size	8	16	4	24	32
Join/Leave Communication Cost(k/s)	$2^6*0.05*64 = 819$	$2^{16}*0.05*256 = 838861$	$2^{16}*0.05*4 = 13107$	$2^{16}*0.05*8 = 26214$	$2^{16}*0.05*1024 = 3355443$

六、結論

本文提出了 Crown，在格網計算的概念中建構分散式的資源搜尋協定，Crown 為完全分散式的架構，具有很高的擴充性以及良好的繞送效率，並加上了群組的與複本的設計，進一步強化資源搜尋的速度及可靠可用性。

Crown 目前的設計環境，都是假設節點是正常且無惡意的運作，當面對惡意的節點（Malicious Node）的攻擊時，可能產生的各種安全性問題，如在索引值值的存取加上權限管理的機制、受惡意攻擊後的錯誤回復等，都是將來可以發展的方向。

七、參考文獻

[1] Foster, I .”*Grid computing and web services a natural partnership.pdf*” Parallel, Distributed and Network-based Processing, 2002. Proceedings. 10th Euromicro Workshop on , 2002 Page(s): 3 –3.

[2] I. Foster, C. Kesselman. *“Computational Grids” , Chapter 2 of "The Grid: Blueprint for a New Computing Infrastructure"*, Morgan-Kaufman, 1999.

[3] I. Foster, C. Kesselman, J. Nick, S. Tuecke , *“The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration”*. Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002.

[4] Germn Goldszmidt, Yechiam Yemini. *“Evaluating Management Decisions via Delegation”*. IFIP International Symposium on Network Management, April 1993.

[5] http://www.w3.org/2002/ws/

[6] http://www.uddi.org/pubs/the_evolution_of_uddi_20020719.pdf

[7] D. R. Karger, E. Lehman, F. Leighton, M. Levine, D. Lewin, and R. Panigrahy, *“Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on theWorldWideWeb,”* in Proc. 29th Annu. ACM Symp. Theory of Computing, El Paso, TX, May 1997, pp. 654–663.

[8] M. Ripeanu, I. Foster , *“A Decentralized, Adaptive, Replica Location Service”*. 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11) Edinburgh, Scotland, July 24-16, 2002.

[9] Napster. http://www.napster.com/

[10] C. Greg Plaxton, Rajmohan Rajaraman, and Andrea W. Richa. “Accessing nearby copies of replicated objects in a distributed environment”. In Proceedings of ACM SPAA. ACM, June 1997

[11] Kavitha Ranganathan, Adriana Iamnitchi, and Ian Foster,**”** *Improving Data Availability through Dynamic Model-Driven Replication in Large Peer-to-Peer Communities”*, Global and Peer-to-Peer Computing on Large Scale Distributed Systems Workshop, Berlin, Germany, May 2002.

[12] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, Scott Shenker**,”***A Scalable Content-Addressable Network”***,** Proceedings of ACM SIGCOMM 2001

[13] RFC 3174 *US Secure Hash Algorithm 1 (SHA1)*

[14] A. Rowstron and P. Druschel, *"Pastry: Scalable, distributed object location and routing for largescale peer-to-peer systems,"* in Proc. 18th IFIP/ACM Int'l. Conf. Distributed Systems Platforms (Middleware), 2001.

[15] Stoica, I.; Morris, R.; Liben-Nowell, D.; Karger, D.R.; Kaashoek, M.F.; Dabek, F.; Balakrishnan,*H* . *“Chord: a scalable peer-to-peer lookup protocol for Internet applications“*, Networking, IEEE/ACM Transactions on , Volume: 11 Issue: 1 , Feb. 2003 Page(s): 17 -32.

[16] The Gnutella Protocol Specification v0.4 Revision 1.2.

[17] B. Y. Zhao, J. Kubiatowicz, and A. Joseph, *"Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing"*. Tech. Rep. UCB/CSD-01-1141, University of California at Berkeley, 2001.