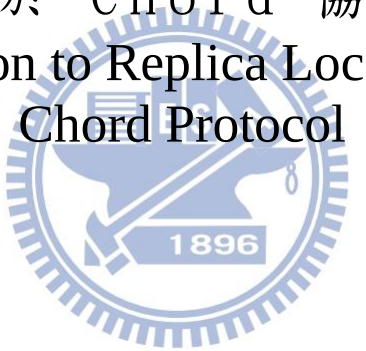
國立交通大學

電機學院電信工程研究所碩士班

碩 士 論 文

應用雜湊函數在基於 Chord 協定的複本位置服務
Applying Hash Function to Replica Location Service Based on Chord Protocol



研 究 生：李俊緯

指導教授：李程輝　教授

中 華 民 國　一　佰 零 二 年 八 月

應用雜湊函數在基於 Chord 協定的複本位置服務

Applying Hash Function to Replica Location Service Based on Chord Protocol

研 究 生：李俊緯　　Student：Chun-Wei Li

指導教授：李程輝　　Advisor：Tsern-Huei Lee

國 立 交 通 大 學

電機學院電信工程研究所碩士班

碩 士 論 文

A Thesis
Submitted to College of Electrical and Computer Engineering
National Chiao Tung University
for the Degree of
Master
in

Institute of Communication Engineering

June 2013

Hsinchu, Taiwan, Republic of China

中華民國 一佰零二 年八月

應用雜湊函數在基於 Chord 協定的複本位置服務

學生：李俊緯　　指導教授：李程輝

國立交通大學電機學院電信工程研究所碩士班

摘　　要

在大型對等系統(Peer-to-peer system, P2P)裡，由於點的新增與失效和存取資料很頻繁，因此系統的負載平衡(Load Balance)與資料可靠性(Reliable)顯得很重要，在之前 Chord 協定針對決定資料放置的位置與有效且快速的搜尋資料做了很好的解決方案。

而在這篇論文裡，我們則是基於 Chord 協定使用雜湊函數結合複本技術，我們稱之為雜湊式複本位置服務，我們使用這個方法來使得系統有良好負載平衡且維持資料的可靠性，事實上，雜湊式複本位置函數不需改變 Chord 核心技術，只需增加維護函式，因此我們可以容易且很快的實現在 Chord 協定上面。

比較相關的複本技術，我們的雜湊式複本位置服務擁有更好的負載平衡與可靠性。

關鍵字：對等系統、雜湊式複本位置服務、負載平衡、分布式系統、搜尋資源服務

Applying Hash Function to Replica Location Services Based on Chord Protocol

student：Chun-Wei Li　　Advisors：Prof. Tsern-Huei Lee

Institute of Communication Engineering
Electrical and Computer Engineering College
National Chiao Tung University

ABSTRACT

In large-scale peer-to-peer system, the load balance and data reliable is very important because node arrivals and departures frequently. In fact, Chord protocol is a powerful solution which is the efficient location of the node that stores a desired data item.

In this paper, we use replication technology with hashing based on Chord protocol, called Replica Location Service with Hashing. We use it to get the great load balance and data reliable in P2P system. In fact, the Replica Location Service with Hashing doesn't change any primary protocol and just adds new maintain function in Chord. We can implement on Chord protocol easily and quickly.

Compared with related replication technology on Chord protocol, the proposed Replica Location Service with Hashing is more balanced and more reliable.

Keywords: peer-to-peer systems, Replica Location Service with Hashing, load balance, distributed systems, resource discovery service

誌　　謝

能完成這篇論文，首先我要感謝我的指導教授李程輝教授，經由教授的耐心指導，教我正確的研究態度與方法，訓練我思考與解決問題的能力，得以完成本篇論文。

再來感謝同一組的夥伴廣煜、鐙標和彥良，平時互相討論研究問題，提出許多非常好的建議與盲點；還要感謝另外一組的夥伴、學長姐和學弟妹，平時的鼓勵與建議。

感謝我的朋友們，一起分攤平時作研究的壓力，讓我得以度過這兩年，也讓這兩年的碩士生活精彩難忘。

接著我要感謝我的父母與弟弟，有你們的關心支持與栽培照顧，並供應我食衣住行，讓我能後顧之憂的專心研究。

最後謹將此論文獻給身邊所有愛我的人及我愛的人。

2013/08　李俊緯

目　　錄

圖　目　錄

表　目　錄

第一章

簡介

1.1 簡介

對等網路(peer-to-peer, P2P)是個非中心伺服器控制的網路，每個點都運行自己一套軟體，每個點既是使用者也是伺服器，任何一個點不一定能直接找到其他點，需藉由自己連接的用戶群來進行找尋，對等網路的節點能遍及所有網路，在檔案分享、平行運算、資料儲存等得到廣泛的應用。

在現今巨量資料的時代，大量使用者存取與產生資料，很多時候都會利用對等網路技術提升系統上的效能和達到健壯性與資料的可靠性，而在對等網路下，會透過在多個節點上複製資料達成資料的可靠性與防止資料遺失，也可以達到負載平衡的效果，因此複本技術便顯得很重要。

Chord 協定[1]（Chord Protocol）是對等網路的四大基礎協定之一，由 MIT 提出，其他三個分別是 CAN [2]、Tapestry [3]、Pastry [4]，Chord 協定主要提供一個在對等網路裡快速找尋資料的演算法，原本 Chord 協定是不關心資源是如何儲存的，只著重於資料的找尋；Chord 協定透過一致性雜湊函數[5]（Consistent hashing）針對每個點給予一把金鑰，且每筆金鑰並映射到相同空間上，為了確保金鑰不會重複，Chord 協定使用 SHA-1 [6]做為雜湊函數，這些金鑰視作整數並照順序排列頭尾相連，因此可以視為一個虛擬的環狀結構，使用 Chord 協定產生出來的對等網路為結構化對等網路，而使用雜湊函式會導致幾種特性：每個點之間不會擁有相同的金鑰、資料也會經過雜湊函數產生獨立的金鑰映射到相同空間，且當每個點加入與離開時只需移動少量的金鑰；除此之外，Chord 協定擁有自己的搜尋演算法，只需要部分點的資訊，便能完成找尋資料位於哪個

點的功能，在所有點數為 N 的情況下，每個點之需要得知$O(logN)$個其他點的資訊，且找尋資料的訊息也只會經過$O(logN)$次的跳動。

但是基本的 Chord 協定無法支援複本儲存，假如有資料複製成兩份，經過一致性雜湊函數會產生相同的金鑰，導致同樣的資料儲存在同樣的點上，當點失效資料等於遺失，並無法達成對等網路裡的增加資料可靠性與系統健壯性，因此提出了 Globus 複本位置服務[7](Globus Replica Location Services, RLS)的概念，每一筆資料會做數位層與實體層的映射，產生一個目錄稱為本地複本目錄(Local Replica Catalogs, LRC)，實體層的意思是指實際儲存位置，有可能資料庫位置、檔案系統位置等等，數位層就是指此檔案的名稱，檔案名稱經過雜湊函式出來的值就稱為金鑰，每個點再儲存這些本地複本目錄，就可以解決複本儲存的問題。

而每個點只儲存本地複本目錄是不夠的，萬一點失效了，雖然資料沒有遺失，但目錄不見了便無法找到資料是儲存在哪，因此也算是間接導致資料遺失，所以此篇論文便是在討論如何儲存本地複本目錄的複本，使他能跟Chord協定結合，增加負載平衡、搜尋效能、資料可靠性與系統的健壯性。

此篇論文的結構為：在章節二我們會討論相關背景知識，例如：Chord協定、Globus複本位置服務等等，還有定義我們所遇到的問題；在章節三我們會詳細解說雜湊式複本位置服務的建立與對Chord協定產生的影響；在章節四我們會列出所跑的模擬圖與原方法的優劣；在章節五會總結一下我們的結論與探討未來可能的面相。

第二章

研究背景與問題描述

2.1 問題描述

2.1.1 負載平衡

在現今網路服務下，通常建置好幾台伺服器來分散使用者需求，以免瞬間龐大的使用者需求導致伺服器回應過慢或者無法回應，如圖1所示，當很多使用者想要去存取資料A的時候，這時候資料A就是所謂的熱點(Hot Spot)，這時我們把資料A儲存在資料庫A跟資料庫B的地方並經由演算法讓使用者需求分散在兩台伺服器，這就是所謂的負載平衡，當兩台伺服器分配越平均就有越好的負載平衡，而最後我們會討論本篇論文擁有最好的負載平衡。

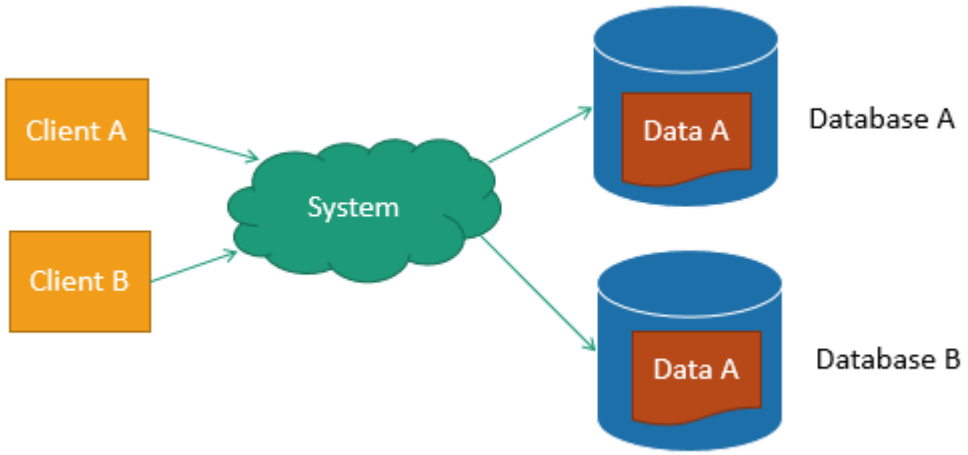


圖 1　系統服務架構

2.1.2 複本技術與資料可靠性和系統健壯性

當複製資料且放在許多台伺服器上面時，同時也增加資料的可靠性，不會因為一台伺服器失效而導致資料遺失，還可以經由切換伺服器就可以達到服務不中斷，我們稱之為資料的可靠性與系統的健壯性，在現今網路服務下越來越顯得重要，而決定資料要怎麼複製與放置，便稱為複本技術，本篇論文主要討論一個新的複本技術並討論其資料可靠性。

2.2 Chord協定

2.2.1 一致性雜湊函數

一致性雜湊函數使用的雜湊函數為SHA-1，對每個點的IP經由SHA-1產生一把長度為m位元的金鑰（Key, K）且把這金鑰當作點的標識符（Identifier, id），同理對每個檔案或資料經由SHA-1也產生一把長度為m位元的金鑰，而且m要夠大使得產生出來的金鑰彼此碰撞的機率很小，而m是設定為160位元，這些金鑰與標識符依據順時鐘從小排到大的方式排列且0與$2^m - 1$相連，形成一個環狀，在環上我們對各個金鑰可以以順時鐘找到離他最近的標識符，而擁有這個標識符的點我們稱之為此金鑰的後點（Successor），而檔案與金鑰都儲存在各自金鑰的前點上；反之，對各個金鑰以逆時鐘會找到離他最近的標識符，而擁有這個標識符的點我們稱之為此金鑰的前點（Predecessor）。

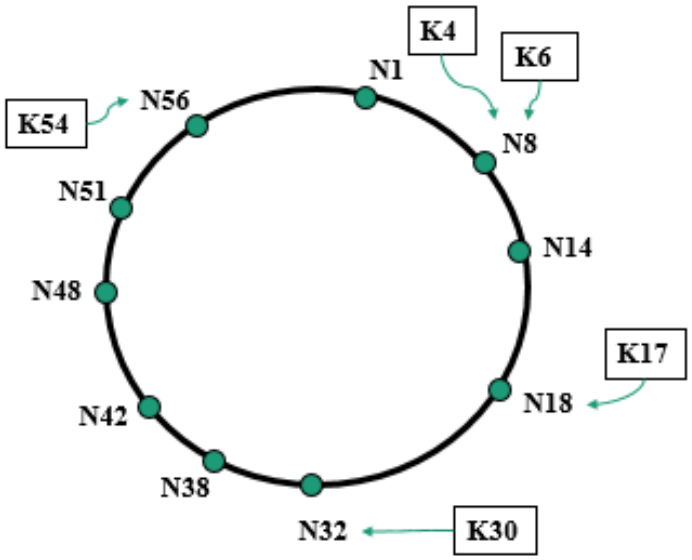


圖 2 一致性雜湊函數上點與金鑰的關係

如圖2所示，K4和K6的檔案與資料儲存在N8，K17的檔案與資料儲存在N18，K30的檔案與資料儲存在N31，K54的檔案與資料儲存在N56，在Chord協定裡也是如此儲存檔案與資料，我們稱之為Chord環（Chord Ring）。

依據這種儲存金鑰的方式，當點刪除時，也只需移動部分檔案與資料，例如圖3刪除N21的時候，只需移動K17的檔案與資料到N32即可；同理當點增加，也只需移動部分檔案與資料，例如圖3新增N4，只需把K4的檔案與資料移動到N4即可。

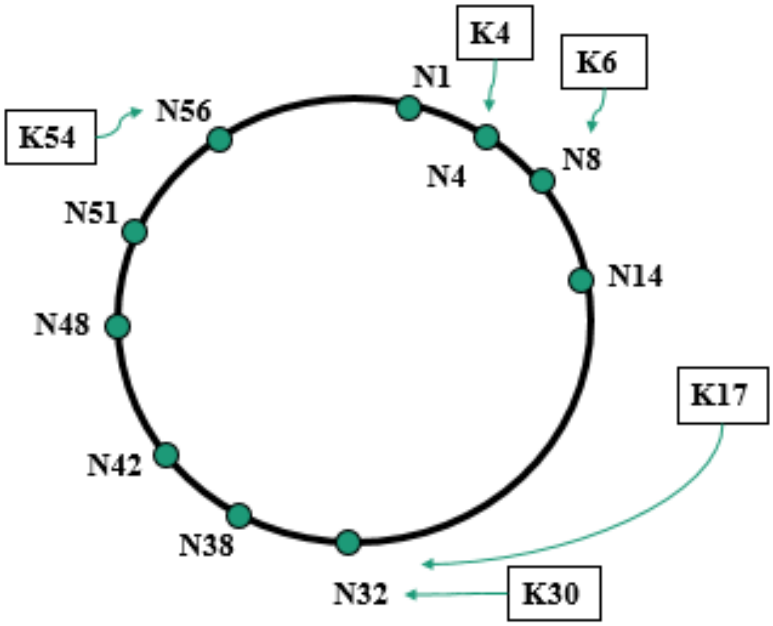
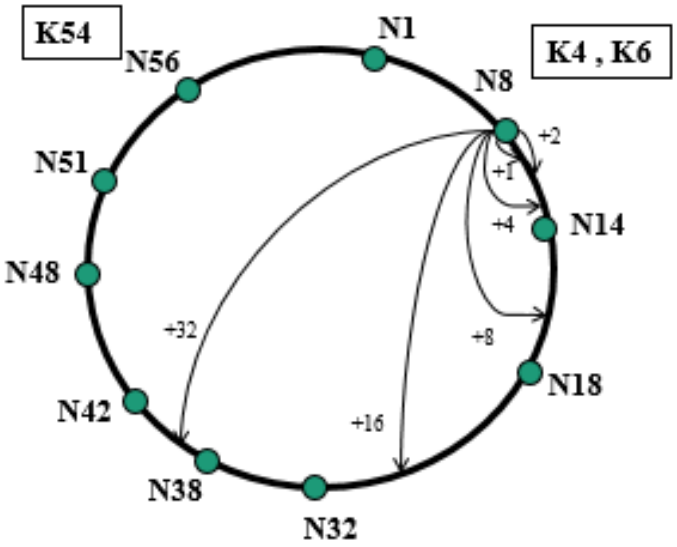


圖 3　點的新增與刪除對資料的影響

2.2.2 延展性的金鑰搜尋方法

決定儲存金鑰的方式之後，接下來就是搜尋金鑰的方式，最原始的方式是以順時針方向對各個點做搜尋，故跳躍數為$O(N)$，效率較差，有人提出記錄每個點的位置就可以達到最小跳躍數，但儲存的表則會非常巨大，加上對等網路點的數量通常很多，因此不符合效益，後來到 Chord 協定提出了延展性的金鑰搜尋方式，儲存部份點的資訊，但卻能增加搜尋速度。



Finger table at N8	
K8+1	N14
K8+2	N14
K8+4	N14
K8+8	N18
K8+16	N32
K8+32	N42

圖 4　指向表的建立

Chord 協定建置了一個表，此表儲存著部分金鑰所儲存的位置，金鑰為 $id + 2^i$，因為儲存的路線圖像手一樣很多手指，因此稱為指向表（Finger Table），例如圖 4，N8 的指向表依序儲存 K9、K10、K12、K16、K24 和 K40，分別在 N14、N14、N14、N18、N32 和 N42 的點上，而指向表儲存的點我們稱之為指向點（Finger Node）。

圖 5 為使用指向表做延展式金鑰搜尋的例子，我們在 N8 上搜尋 K54，先查找 N8 的指向表發現 K54 超過 K40，因此至少位於 N42 之後，便跳到 N42，在查找 N42 的指向表發現 K54 位於 K50 跟 K60 之間，由於 N42 並不知道 N1 跟 N51 之間有無其他比 K54 還大的點，如果跳到 N1 便會找不到金鑰，像此例中間便有個 N56，因此跳到 N51，在查找 N51 的指向表發現 K54 位於 K53 跟 K55 之間，所以跳到 N56，最後便在 N56 上找到 K54；因為指向表特性的緣故，故搜尋金鑰的平均跳躍數為

$$\frac{1}{2}\ log_2(N) \qquad (2.1)$$

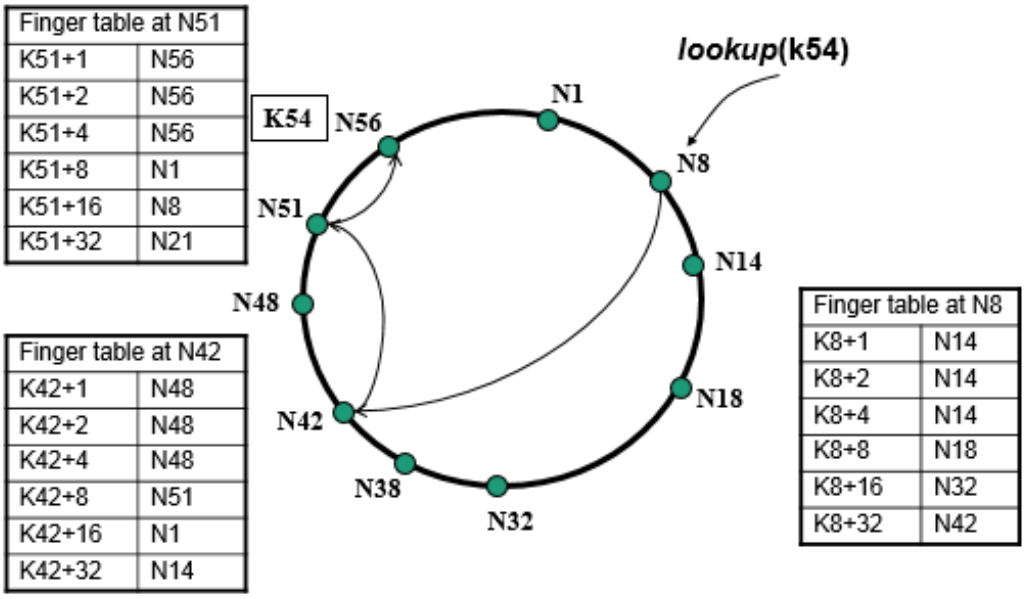


圖 5　使用指向表的延展式金鑰搜尋

2.3　Globus複本位置服務

在對等網路的環境下，為了避免單點失效和增加效率，都會增加多個可供存取的複本，而Globus複本位置服務是管理複本的其中一種技術，他提供了有規則性的註冊和查找資料的複本。

複本位置服務主要分成兩個部份，第一個為本地複本目錄(Local Replica Catalogs，簡稱LRCs)，負責維護數位檔名對物理上儲存位置或儲存系統的對映，例如：檔案 A.txt 儲存在 http://140.113.1.1:10，A.txt - http://140.113.1.10:10 便是檔案A.txt其中一個LRC ；第二個為複本位置索引（Replica Location Indices, RLIs），負責把本地複本目錄製成索引，再根據查找的檔案給予不同的目錄，通常索引裡面包含著一個或多個目錄，例如圖6所示為N8上的索引，裡面分別有K4跟K6的目錄，我們稱為對等網路之下的複本位置索引（Peer-to-Peer Replica Location Indices, PRLIs）。

圖 6　N8 的複本位置索引

我們可以把這概念簡化成階層式的架構，因為當引入複本技術後，每個索引可能儲存數個目錄，每個目錄也可能被數個索引所儲存，如圖7。

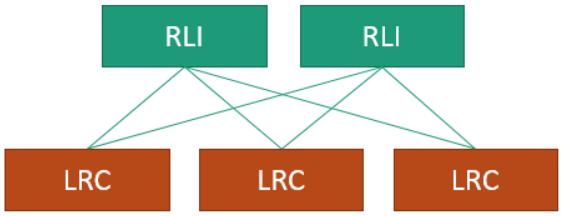


圖 7　本地複本目錄與複本位置索引的階層式關係

Globus複本位置服務便把這應用在Chord協定上，讓Chord協定能處理資源儲存的問題，如圖8所示N8索引儲存K4和K6的本地複本目錄，然後本地複本目錄再指向實際上儲存的位置或檔案系統，此舉可以解決Chord協定無法儲存複本的問題，但這樣的架構依然有單點失效的問題，例如N8失效的話，關於K4和K6的目錄會遺失，這樣照樣找不到K4和K6的檔案位置，因此Globus複本位置服務提出在多個後點儲存複本，稱為後點可適應性複本技術（Adaptive Replication on Successor Nodes）。

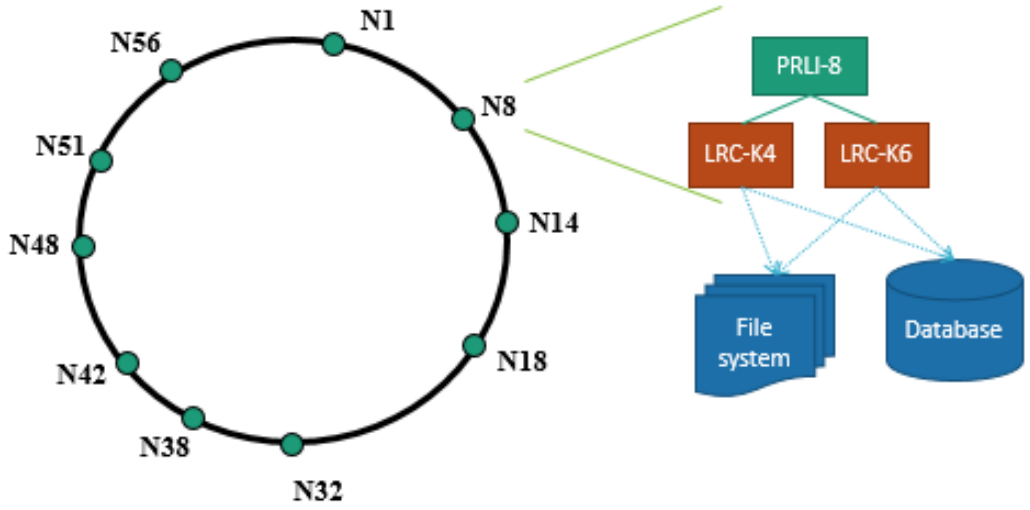


圖 8　在 Chord 協定之下的複本位置服務

後點可適應性複本技術就是依據需求在數個後點儲存複本，例如圖9以N8為例，為了防止單點失效，N8在自己的後點上儲存了K4和K6的目錄，因為當N8失效時，依據一致性雜湊函數儲存規則，K4和K6要儲存在N14，而此種複本技術便可讓當N8失效時，金鑰的儲存符合一致性雜湊函數的儲存規則，不用再花額外的資料遷移。

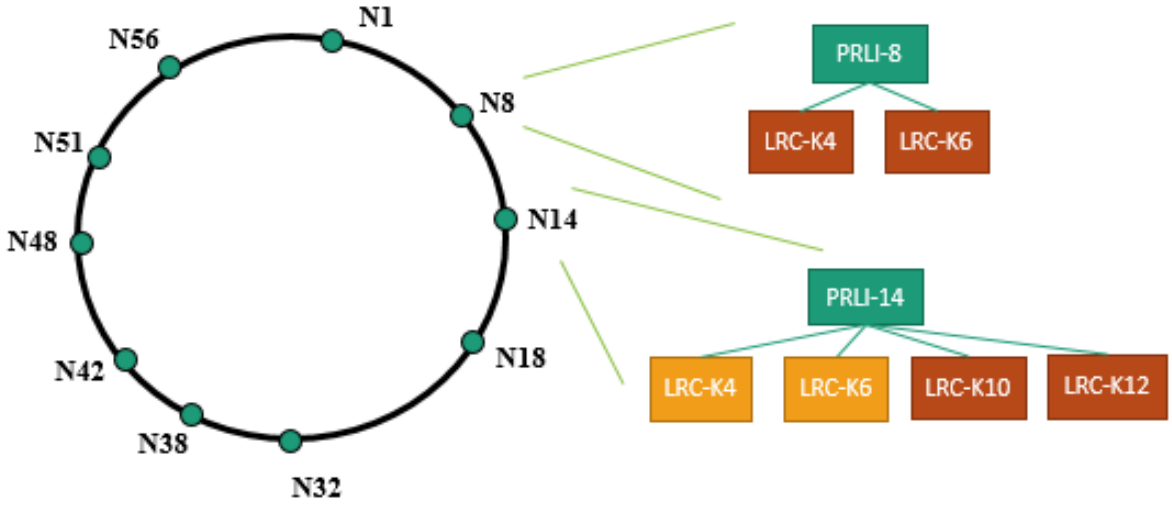


圖 9　複本儲存在後點示意圖

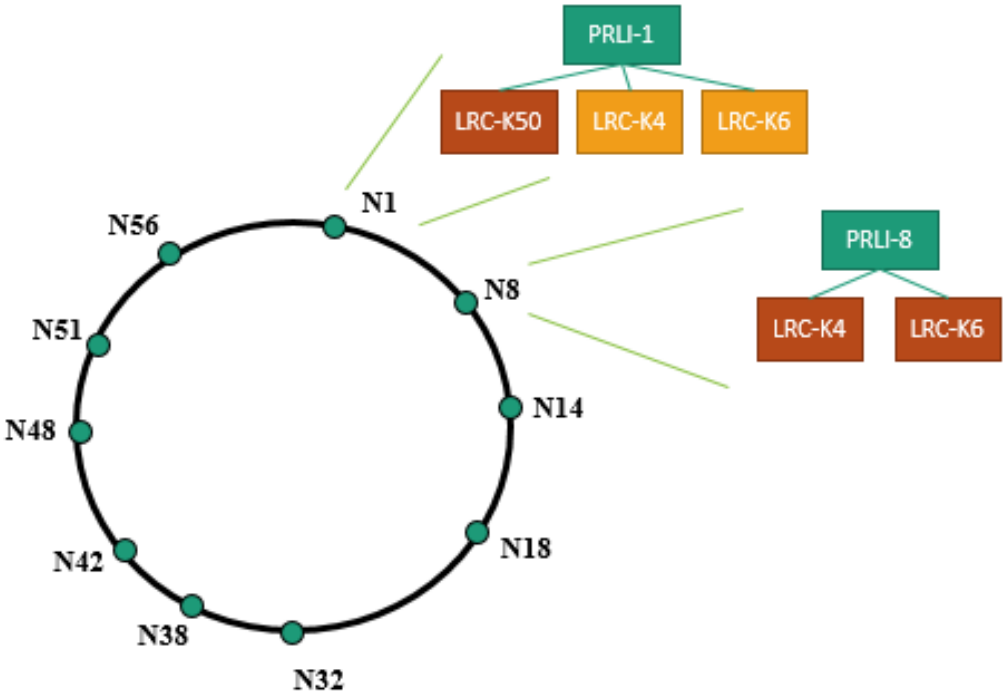


圖 10　複本儲存在前點示意圖

由於 Chord 延展性搜尋方法（章節 2.2.2）的特性，越接近目標點跳躍的距離越短，因此有很高的機率會跳躍到目標點的前點上，所以 Globus 複本位置服務為了增進系統的負載平衡，提出了在多個前點儲存複本，已達到負載平衡的效果，例如圖 10 為例，N8 為了分散自己的負載，便複製一份 K4 和 K6 的目錄到前點 N1 上。

通常來說分散後的負載不一定是均衡負載，因此在章節四我們會討論負載平衡與資源利用的公平性；除此之外，儲存在後點的優點是能增加資料的可靠性，但負載平衡的能力就很低，因為跳躍到後點的機率是比較小的，而儲存在前點的優點是有比較好的負載平衡，但資料可靠性就會下降，因為當 N8 失效時，N1 的 K4 跟 K6 目錄並不會更新到 N14 上，導致搜尋時，會有機率找不到 K1 跟 K6。

第三章

雜湊式複本位置服務

3.1 雜湊函式

之前的複本位置服務是儲存在原本點的前一個點上或者是後一個點上，在有部分資料存取頻繁的的情況下，負載平衡的效果不好，且資料的可靠性並不那麼裡想，而本篇論文是在探討如何取得良好的負載平衡與資料的可靠性。

因此本篇論文使用一個雜湊函式，如圖11所示，把原本資料的金鑰在經過函式產生多把金鑰，把這些金鑰當作複本的金鑰，稱作複本金鑰（Replica Key, RK），然後複本依據複本金鑰決定他們儲存的位置，跟原本決定資料儲存位置的方式一樣。

圖 11 產生各種 Key 的流程

雜湊函式的公式如下：

$$RK_j = \left(Key + j \times \frac{n}{r} \right) modulo\ n, 0 < j < r, r = 2^i, 0 < i \leq m \qquad (3.1)$$

又 n = 2^m，故又可簡化成

$$RK_j = \left(Key + j \times 2^{m-i} \right) modulo\ n, 0 < j < 2^i, 0 < i \leq m \qquad (3.2)$$

r是複本加上正本的數量即資料總數，其數量為2的指數，最多為2^m即所有點都存有複本和正本，這樣金鑰的分佈方式在維護時可以使複本位置跟指向表有相關，即利用指向表就可以搜尋到儲存複本的位置，不用額外的儲存空間來記錄複本位置（章節3.2）；而在搜尋時，可以經由運算得知所在的點與哪一個複本最近(章節3.4.1)。

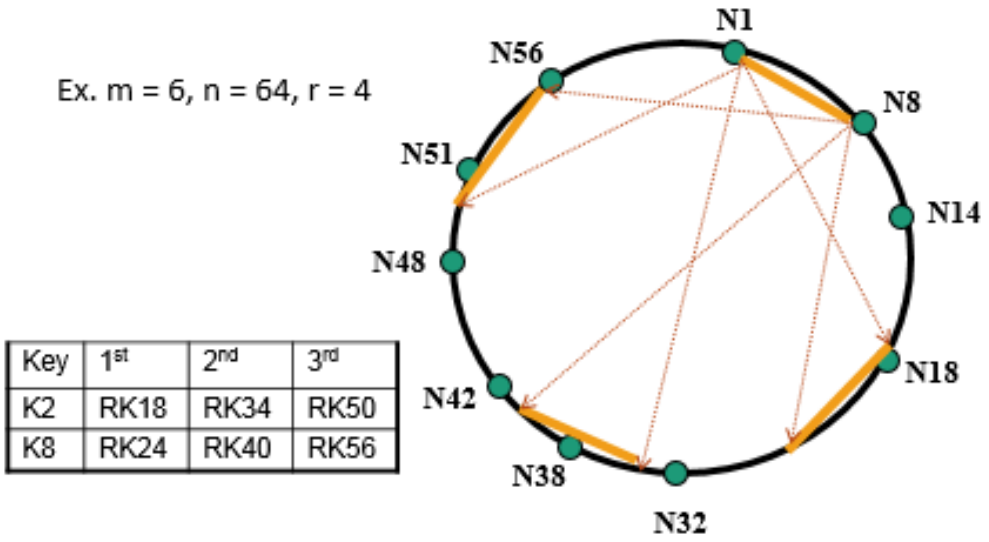


圖 12　在 N8 上所儲存資料的複本分布

而複本的分布方式會呈現Chord環被切為r等份的形式，如圖12所示，以N8為觀點、$r = 4$為例，N8儲存著K2～K8的正本，第1個複本範圍在RK18~RK24位置會在N18、N32，而第2個複本範圍在RK34~RK40位置會在N42、N38，而第三個複本範圍在RK50~RK56位置會在N51、N56，如此均勻分散的特性使得在每個點均勻分布之下，能提供良好的負載平衡。

3.2　維護複本函式

當決定好複本放置的位置後，另一個重點就是如何維護他們，使在有限的點失效後還能保持一定數量的複本，並建立恢復複本機制，增加資料的可靠性。

依據公式(3.2)所產生的複本金鑰，事實上與指向表有所關聯，在章節2.2.2我們可以看到指向表的生成跟點本身的標識符有關，而標識符另一方面來看也是一把金鑰，因此我們可以透過公式來表示指向表與複本金鑰的關係，經由觀察公式(3.2)和圖4得知，RK_j會位於$finger[m-1-log(r)+j]$的指向點或者指向點的前點上。

如圖13所示N8為例，指向表是由N8的標識符產生的金鑰K8所生成的，而本身也儲存著金鑰為K8的正本，因此當$r = 2$時，$RK_1 = 8 + 2^5 = 40$，K8正本的複本恰好就是位於$finger$[6]的指向點；K7正本的複本$RK_1 = 39$也是儲存於$finger$[6]的指向點，依此類推，複本將會儲存在$finger$[6]的指向點或指向點的前點上，因此維護複本只需從指向表尾端

的指向點開始尋找，就能簡單的找到儲存複本的點。

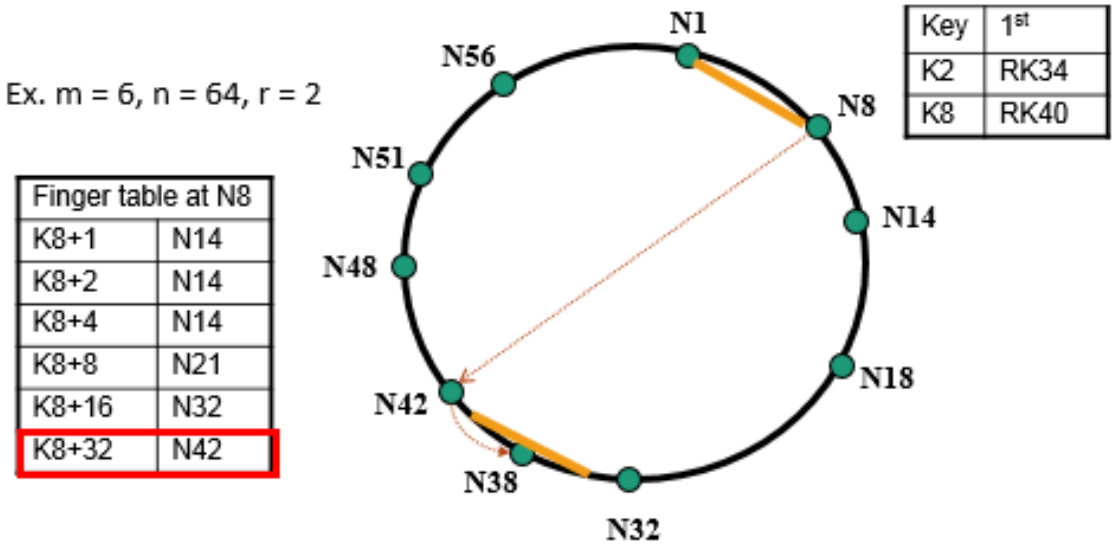


圖 13 當 *r* = 2 時，N8 維護複本示意圖

當 *r=2* 時，因為(3.2)公式，故 N8 知道要去 *finger*[6]的指向點（N42）或者他的前點（N38）維護複本，同理當 *r=4* 時，N8 會去 *finger*[5]跟 *finger*[6]的指向點（N32、N42）或者他們的前點（N38、N18）維護複本，當然依據點分布的不同，有可能往前找很多前點，但最多就是跳回自己本身。

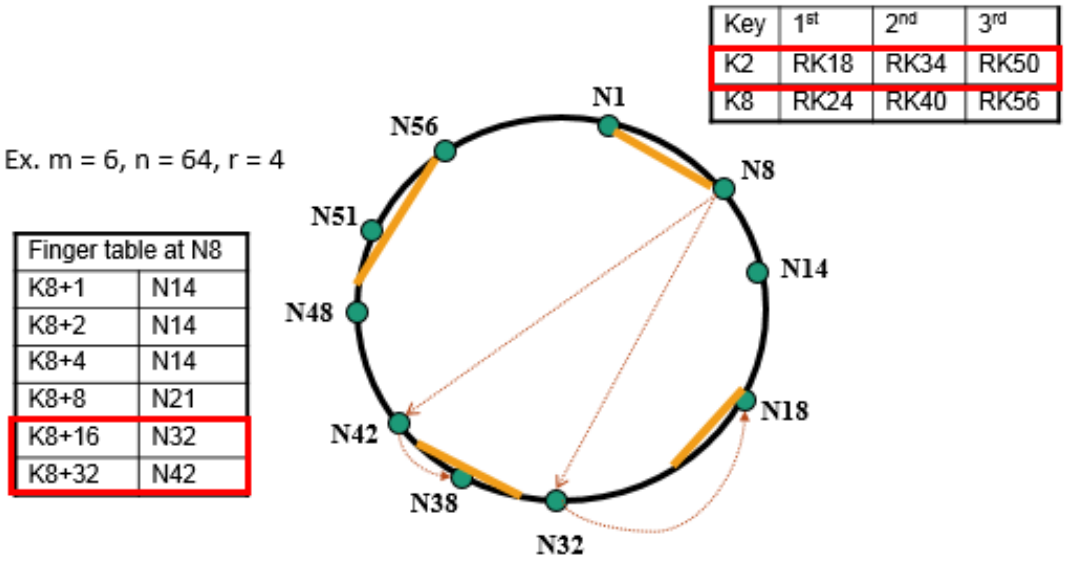


圖 14 當 *r* = 4 時，N8 維護複本示意圖

如果單純只是這樣那會有些複本是無法維護的，例如圖 14 以維護 K2 為例，N8 會藉由維護副本函式，找到 N18 跟 N38 維護 RK18 跟 RK34 的複本，而 RK50 由於 N8 無

法直接經由指向表找到，因此需設計由其他的點來幫忙維護，如圖 15 所示，N18 經由運算 K2 的複本金鑰 RK18 算出其他的複本金鑰，然後再藉著維護副本函式找到 N38 跟 N51 去維護 RK34 跟 RK50 的複本；同理 N38 藉由維護副本函式找到 N51 跟 N8 去維護 RK50 的複本跟 RK2 的正本。

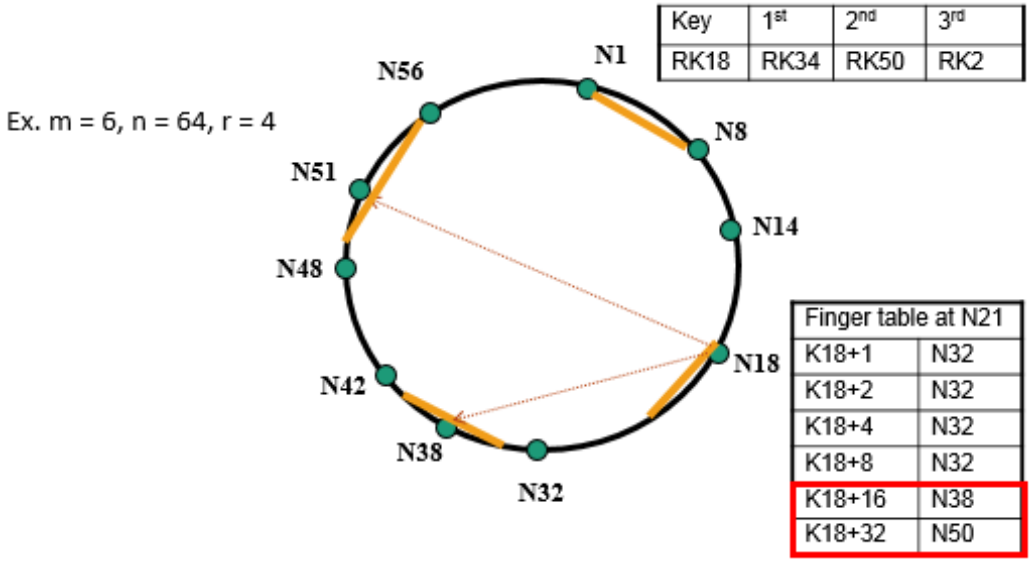


圖 15　當 $r = 4$ 時，N21 維護 K2 正本示意圖

因此每一個複本是由許多點所共同維護，藉此增加資料的可靠性，當部分的點失效時，並不會造成資料遺失，而且藉由維護複本函式可以自我恢復遺失的資料，假設現在所有金鑰都被使用且其中一點儲存 f 筆資料，則在雜湊式複本位置服務裡這一點維護複本的需求數：

$$f \times log_2 r \times r \tag{3.3}$$

而 Globus 複本位置服務在這一點維護複本的需求數：

$$f \times (r - 1) \tag{3.3}$$

表 1 顯示每個複本被維護的點數與資料總數的關係，可以看到當資料總數越多，雖然雜湊式複本位置服務所需的維護複本需求數增加，但每個複本被更多的點維護，因此增加資料自我恢復力和可靠性；反觀 Globus 複本位置服務維護複本需求術也增加，但每個複本事實上沒有被更多的點維護，並不會對資料的自我恢復力產生影響。

表 1　資料總數與維護複本之關係

資料總數	雜湊式複本位置服務		Globus 複本位置服務	
	每個複本被維護的點數	每個點維護副本的需求數	每個複本被維護的點數	每個點維護副本的需求數
2	1	f x 1 x 2	1	f x 1
4	2	f x 2 x 4	1	f x 3
8	3	f x 3 x 8	1	f x 7
⋮	⋮	⋮	⋮	⋮
2^i	i	f x i x 2^i	1	f x $(2^i - 1)$

```
//run periodically to maintain n's replica
n.maintain_replica()
        if(next > i)
                next = 1;
        While(key = get the maintaining file's key)
                replica_key = n.get_replica_key(key, next);
                original_key = get the file's original key;
                n' = n.closet_succeeding_finger_node(replica_key, next)
                n" = n'.find_replica_node(replica_key);
                n maintains files with n" and both files' original_key must be the same;
        next = next +1;
//ask node n to get replica key
n.get_replica_key(key, jth)
        replica_key = (key + j * n / r) modulo n;
        return replica_key;
//search the local table for the closet succeeding finger node of replica key
n.closet_succeeding_finger_node(replica_key, jth)
      return finger[m – 1 – i + j];
//search the node which contains the replica
n.fine_replica_node(replica_key)
        if( replica_key ∈ (n.predecessor, n] )
                return n;
        else if( n.predecessor > n )
                if( n ∈ [replica_key, n.predecessor) )
                        return n;
                else
                        n' = n.predecessor;
                        return n'.fine_replica_node(replica_key);
        else
                n' = n.predecessor;
                return n'.fine_replica_node(replica_key);
```

圖 16　維護複本函式的虛擬程式碼

而圖16是維護複本函式的虛擬程式碼，為了維護資料可靠性需要定期去執行*n*.**maintain_replica**()，首先取出需要維護資料的金鑰，然後使用此金鑰去運出他的下一個*replica_key*，接著使用*replica_key*執行*n*.**closet_succeeding_finger_node(***replica_key***)**找出可能儲存*replica_key*的指向點，再執行*n*.**find_replica_node(***replica_key***)**去找出實際儲存*replica_key*的點，最後就是檢查此點上有無相同正本金鑰的複本，沒有的話就複製一份傳輸過去。

3.3 金鑰的碰撞

使用了新方法之後，會導致複金鑰的碰撞，如圖17，當$r = 4$的情況下，K4正本運算出來的$RK_2 = 36$會與K20正本運算出來的$RK_1 = 36$產生碰撞，因此搜尋RK36時，會不知真正要找的資料是哪一個。

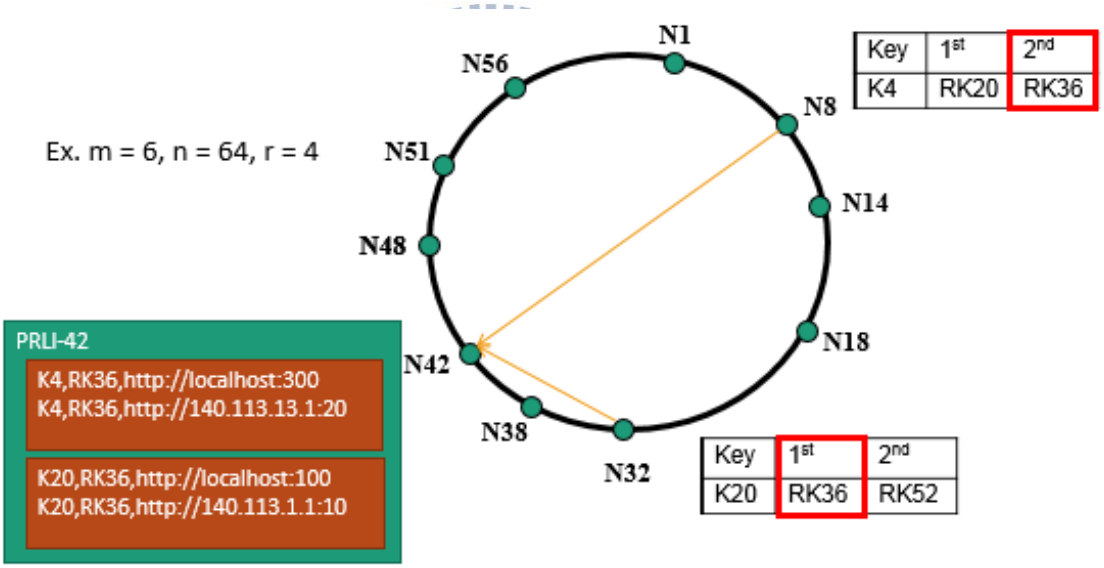


圖 17 K4 的 RK36 與 K20 的 RK36 產生碰撞

為了解決此問題，我們修改了本地複本目錄，多儲存一份正本金鑰，供判別資料的來源，如圖18來自正本K4的複本RK36，多一個欄位儲存K4已標示他是來自正本K4，依此類推，解決金鑰碰撞的問題。

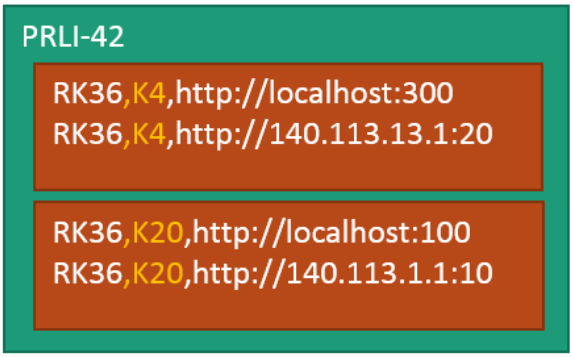


圖 18　本地複本目錄的變動

3.4　在雜湊式複本位置服務下的Chord協定影響

3.4.1 資料的搜尋

原本資料的搜尋是依據查找指向表的方式來做搜尋，方向為順時針的方向搜尋而次數最大為log_2N，而雜湊式複本位置服務下則是在搜尋之前新增了複本金鑰的判斷。

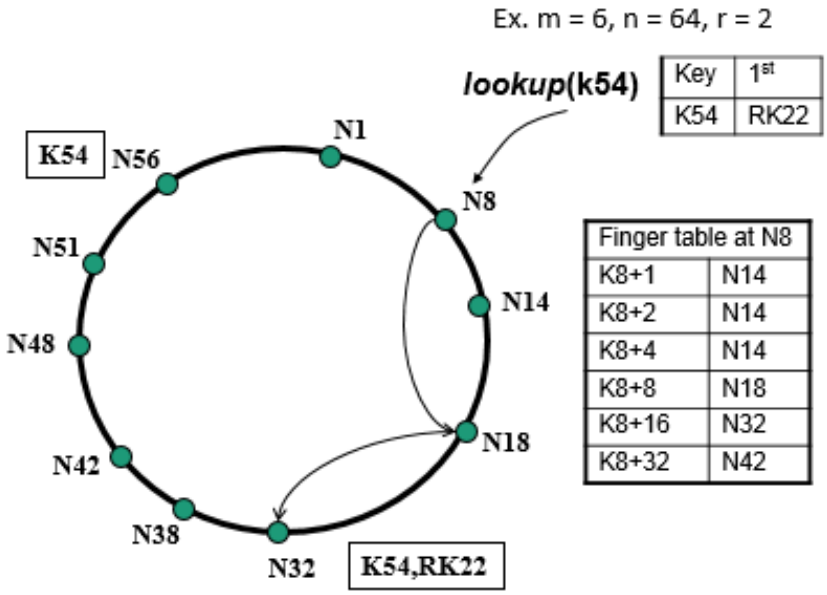


圖 19 基於 Chord 協定在雜湊式複本位置服務下的資料搜尋

如圖19以搜尋K54的正本、$r = 2$為例，在搜尋之前會運算複本金鑰得知為RK22，再去判斷K54跟RK22哪一個以N8為起點順時針來看距離最近，由此可知RK22為距離最近便以金鑰RK22做為搜尋，由指向表得知RK22位於K16與K24之間，因此搜尋到N18，再

由N18得知RK22位於N32，最後在N32上找到RK22複本取得K54的資料。

用此方法可以減少跳躍的次數，假設複本加正本的數目為r，總點數為N，那麼Chord環將被分成r個部分，因此每個部分含有 $\frac{N}{r}$ 個點，所以我們每次使用指向表最多只需跳躍$log_2(\frac{N}{r})$，平均跳躍步數大約為：

$$\frac{1}{2}\ log_2(\frac{N}{r}) \tag{3.3}$$

3.4.2 點的新增

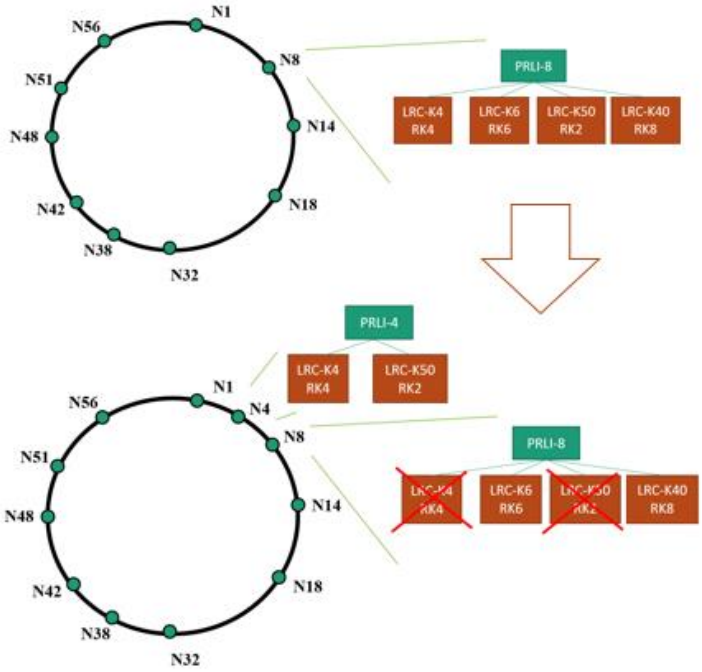


圖 20 基於 Chord 協定在雜湊式複本位置服務下的點的新增

在原本Chord協定之下。以新增前一個點為例，點的新增只需移動部分正本至新的點上；而在Globus複本位置服務下，如果在後x個點有複本，那就需要通知第x個點移除部分複本，同理如果在前y個點有複本，就需要通知第y個點移除部分複本(章節2)；而在雜湊式複本位置服務下，只需移動正本與複本，移動複本的條件與正本一致，檢查哪些金鑰位於新的點的儲存資料的索引值範圍內，而其他的點則不受資料變動的影響。

如圖20新增N4，所以N8裡複本金鑰在RK2到RK4範圍裡的資料將會移動，因此複本

RK4跟RK2將會移動到N4，原本N8裡的RK4跟RK2將會刪除，保持資料儲存的正確性，而N1裡的資料將不會有所變動，被影響的點僅僅為N8。

3.4.3 點的刪除

點的刪除有分成兩種，一種是經由正常程序把資料移到其他點繼續維護，然後再結束點的運作；另外一種則是無預警的點失效，導致資料無法搬移，需要由其他點來恢復，以下就用這一種狀況做討論。

在原本Chord協定下點的失效久等於導致資料直接遺失，因此無法做資料的維護，為原本Chord協定的缺點之一；而在Globus複本位置服務下，當前一個點失效時，如果在後x個點有複本，那就需要通知第x個點新增部分複本，同理如果在前y個點有複本，就需要通知第y個點新增部分複本(章節2)；而在雜湊式複本位置服務下，當有點失效時，只需對後點新增部分複本，而且會經由維護複本函式自行恢復。

如圖21點N4刪除時，N8會接收到N56、N38維護正本K4和N51、N38維護複本RK2，當任一點執行維護複本函式發現資料不存在時，變會自動複製一份，維持資料可靠性。

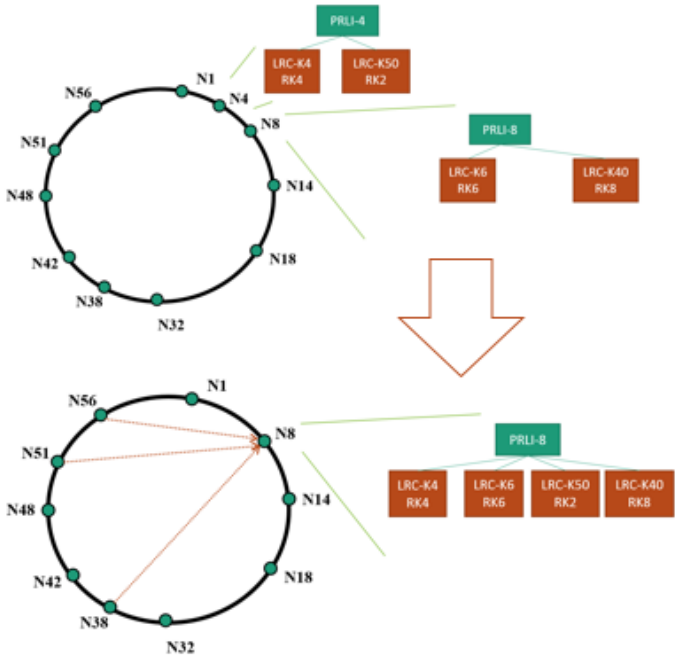


圖 21 基於 Chord 協定在雜湊式複本位置服務下的點的失效

第四章

模擬結果與討論

在這個章節，我們模擬了使用雜湊式複本位置服務的 Chord 協定，討論平均資料搜尋的跳躍數、平均維護複本所需要的次數與負載平衡的分析，並且會與使用 Globus 複本位置服務的 Chord 協定作比較，比較兩者在不同複本數、點數下的狀況，假設的環境參數如下：

1. 網路環境
   a. id 與金鑰長度為： 16 bit
   b. Chord 環所能容納金鑰數或點數： 65536
   c. 點的數量為： 640、1280、2560、5120、10240
2. 正複本相關
   a. 正本和複本總數： 4、8、16
3. 機率分佈
   a. 點：均勻分佈
   b. 各點存取：均勻分佈

使用的開發工具為 Visual Studio Ultimate 2012，作業系統為 Windows 7 64 bit，CPU 為 Intel Core i5-3470 3.2GHz，記憶體為 24 GB 。

4.1 資料搜尋的平均跳躍數

在對等網路之下，由於點的數目龐大，因此如何快速的找到資料在哪，便是一個重要的議題，在此小節會分析我們所提出來方法的資料搜尋速度並與之前方法做比較，討論其優劣。

4.1.1 使用雜湊式複本位置服務的資料搜尋的跳躍數

依據公式(3.3)，我們可以得知總點數與複本數量對平均跳躍數的影響，圖 22 為 r=4 且總點數為2^{12}時資料搜尋跳躍數的機率分佈，經由計算我們可以得知平均跳躍數為 5，由圖可知跳躍數為 5 的時候機率最大，與計算相符合。

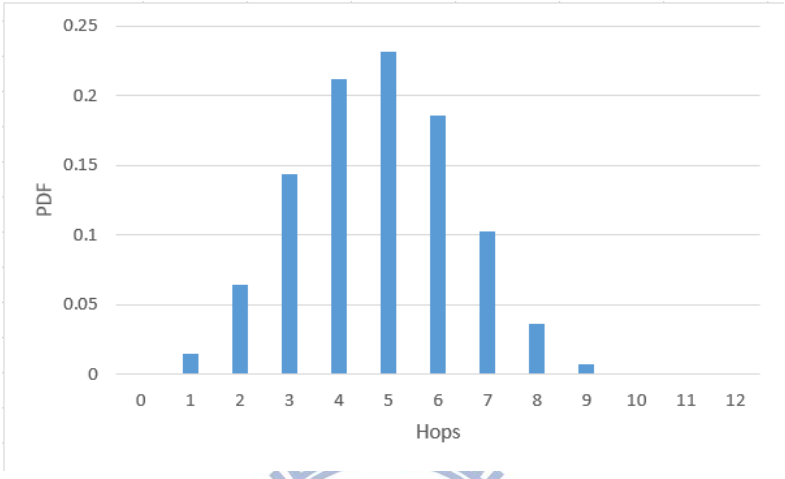


圖 22　r = 4 時，總點數為2^{12}時資料搜尋跳躍數的機率分佈

圖 23 是做 10000 次資料搜尋要求，搜尋金鑰為 50000，搜尋要求是平均分布在存在的點上，然後去計算平均找到目標資料所需的跳躍數，隨著點的數量的增加，我們可以得知平均跳躍次數增加幅度是$O(logN)$，在點的數量很大時，成長是緩慢的，而在對等網路之下，通常是在大量的點互相連接，因此這特性有利於對等網路。

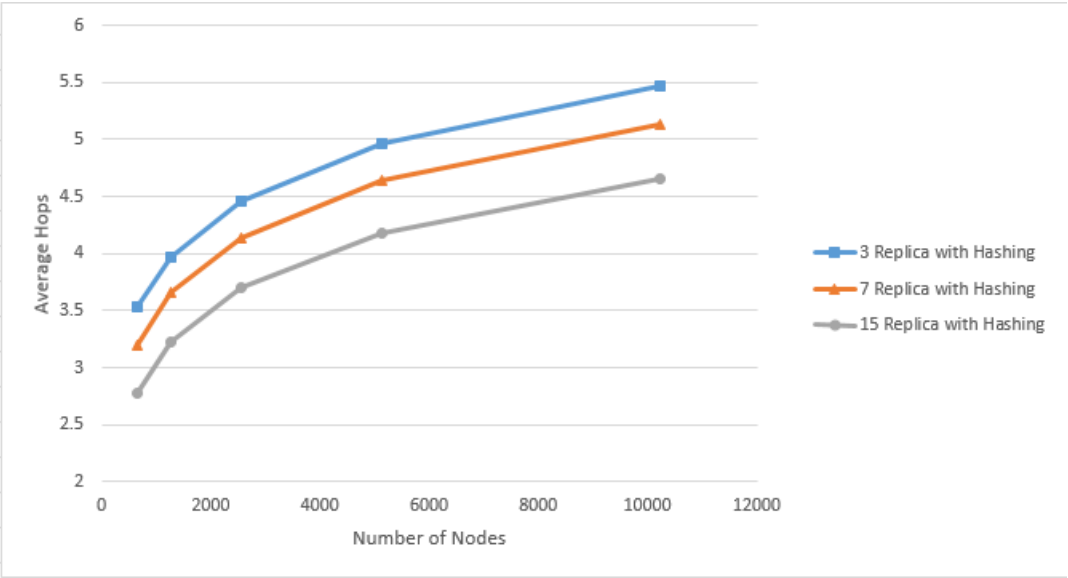


圖 23　點數與複本數量對資料搜尋平均跳躍數的影響

4.1.2 雜湊式複本位置服務與 Globus 複本位置服務的跳躍數比較

接下來我們把雜湊式複本位置服務與 Globus 複本位置服務做比較，圖 24 是在 r = 4 的時候，x 為點數取 log 的座標軸，y 為平均跳躍數，Globus 複本位置服務可以決定要把複本放在前點還是後點，而放的位置不同也會有不同的影響，放在前點可以增加負載平衡(章節 4.2.2)，放在後點則是可以增加資料可靠性，在圖 24 分別跑了四種事件：複本都放在前點、一個複本放在後點、兩個複本放在後點及全部的複本都放在後點，依據章節 2.3，複本都放在前點的複本位置服務資料可靠性最差但搜尋資料速度是最快的，相反的複本都放在後點的資料可靠性最好但搜尋資料速度最快，然後我們在與雜湊式複本位置服務做比較，發現雜湊式複本位置服務資料速度稍微輸給複本都放在後點的複本位置服務，但已經比只放一個複本在前點的好。

而圖 25、圖 26 分別是 r = 8、r = 16 的情況紅色線是雜湊式複本位置服務，可以發現輸複本都放在後點的 Globus 複本位置服務，不過大概跟把四分之一複本數量放在後點的 Globus 複本位置服務差不多，犧牲一點跳躍數但維持資料的可靠性與提供良好的負載平衡。

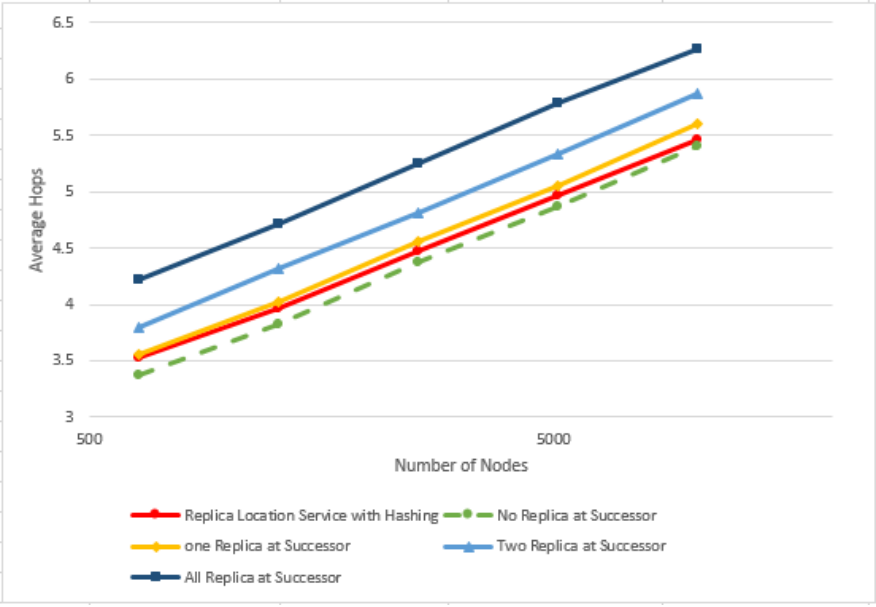


圖 24 r = 4 時，雜湊式複本位置服務與 Globus 複本位置服務下複本儲存在不同位置的跳躍數比較

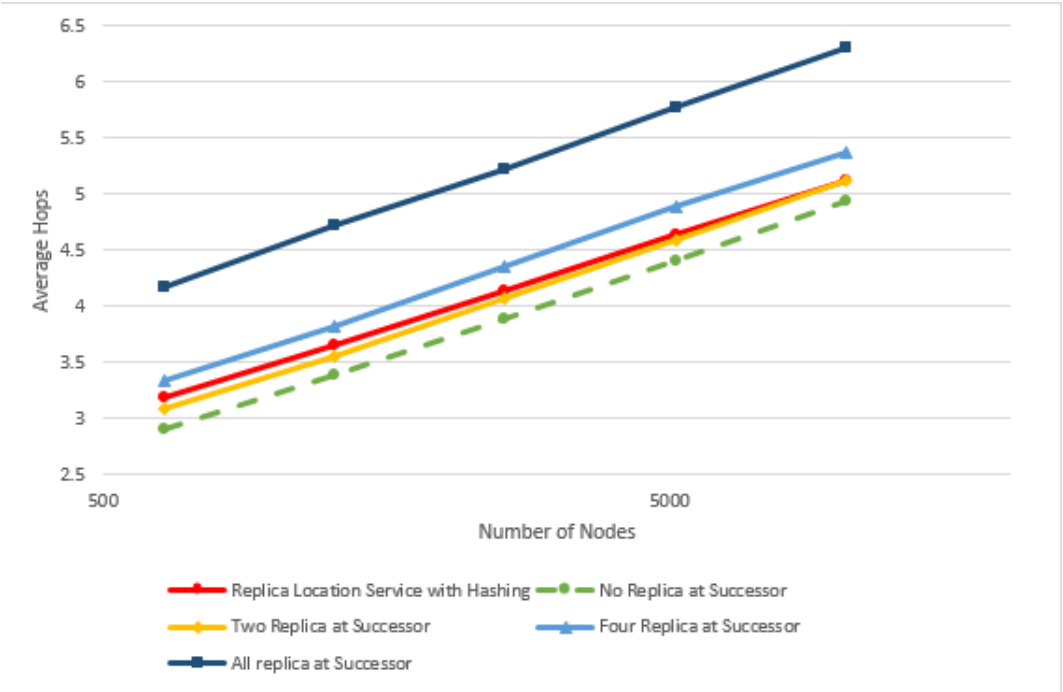


圖 25　$r = 8$ 時，雜湊式複本位置服務與 Globus 複本位置服務下複本儲存在不同位置的跳躍數比較

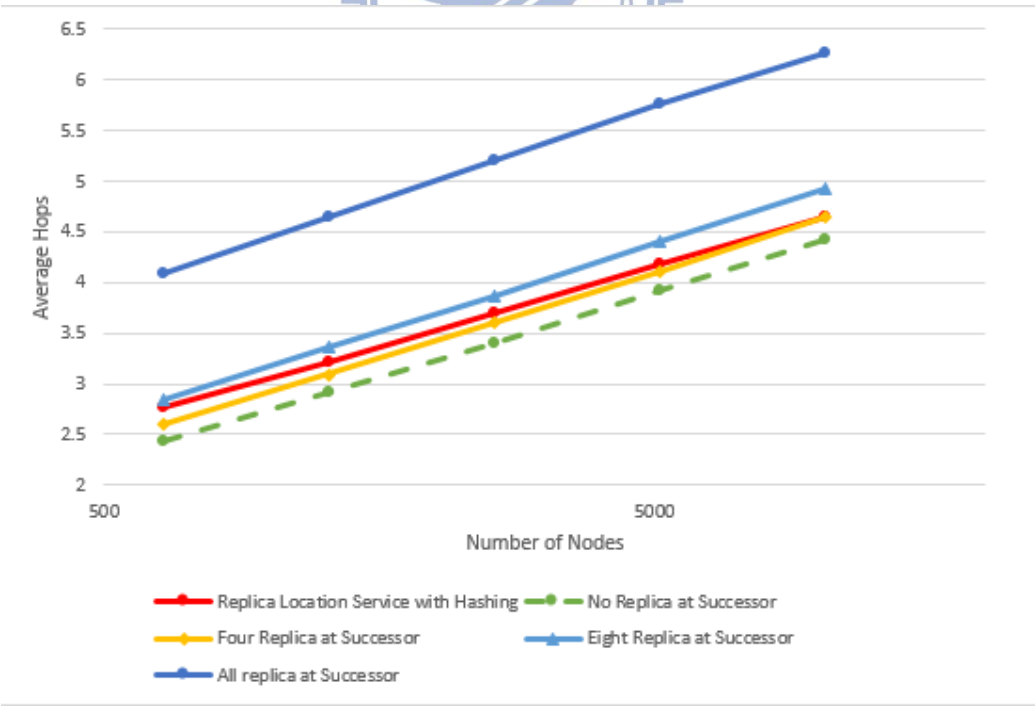


圖 26　$r = 16$ 時，雜湊式複本位置服務與 Globus 複本位置服務下複本儲存在不同位置的跳躍數比較

4.2 負載平衡與資源利用公平性

在對等網路之下，資源能互相分享利用是其一優點，因為能資源共享，所以能做到強大的平行化運算與巨大的儲存空間，但也由於點的數量龐大，因此如何能有效的分配資源便是一個重要的議題，此章節便再討論我們所提出來方法的負載平衡、資源利用的公平性，並與原方法做比較，討論其優劣。

有許多評比公平性的係數例如 Jain’s Fairness index[9][10]與 Kullback-Leibler distance[11]，考慮在描述負載平衡與資源利用公平性上，希望數值能介於 0 與 1，而指數等於 0 時，代表所有負載都集中在一個點上，而當所有負載都平均分散時，指數則等於 1，因此我們使用在無線網路領域，一個資訊理論為基礎的公平性方程式[12]：

$$0 \le \frac{H(S)}{log_2(n)} \le 1 \qquad (4.1)$$

$$H(S) = \sum_i P(X_i) \times i(X_i), \forall i \in \{1,2,\dots,n\} \qquad (4.2)$$

$$i(A) = log_2 \frac{1}{P(A)} = -log_2 P(A) \qquad (4.3)$$

*i(A)*為 Shannon 所稱的資訊本體（Self-information），來表示事件 A 發生時所包含的資訊多寡；*H(S)*為平均資訊本體（Average Self-information），Shannon 定義為熵（Entropy），來表示集合 S 裡會發生事件$\{X_1, X_2, \dots, X_n\}$的資訊本體期望值，依據資訊理論，當所有事件發生的機率都相同時$P(X_1) = P(X_2) = \cdots = P(X_n) = \frac{1}{n}$，則 *H(S)*會有最大值$H(S) = log_2(n)$，當其中一個事件百分之百發生的時候，則 *H(S)*會有最小值$H(S) = 0$，因此換成負載平衡的觀點來看的話，當所有的點都均衡負載的時候，公平性也會隨之上升，而當所有負載集中到同一點的時候，公平性則是最差，故可以利用這個公式來評估整體網路的負載平衡。

4.2.1 雜湊式複本位置函數的負載平衡與公平性

依據公平性公式（4.1）分別在總點數為 640、5120、10240 的情況下，做 100000 次資料搜尋要求，搜尋金鑰為 50000，搜尋要求是平均分布在存在的點上，然後比較不同的資料總數（正本加上複本數量）4、8、16 的公平性，如圖 27 所示，可以看到資料總數增加時，公平性會上升，因為當資料總數越多的時候，負載會越分散到多個點上，比較不容易受到局部擁擠或分佈不均的影響；另一方面，也可以看到當總點數增加的時候，公平性也會上升，道理相同，總點數越多越不容易受到局部擁擠或分佈不均的影響。

在對等網路之下，總點數通常是很多，只要在點分布是平均且平均存取每個點，在對等網路之下可以得到很好的負載平衡與資源利用的公平性。

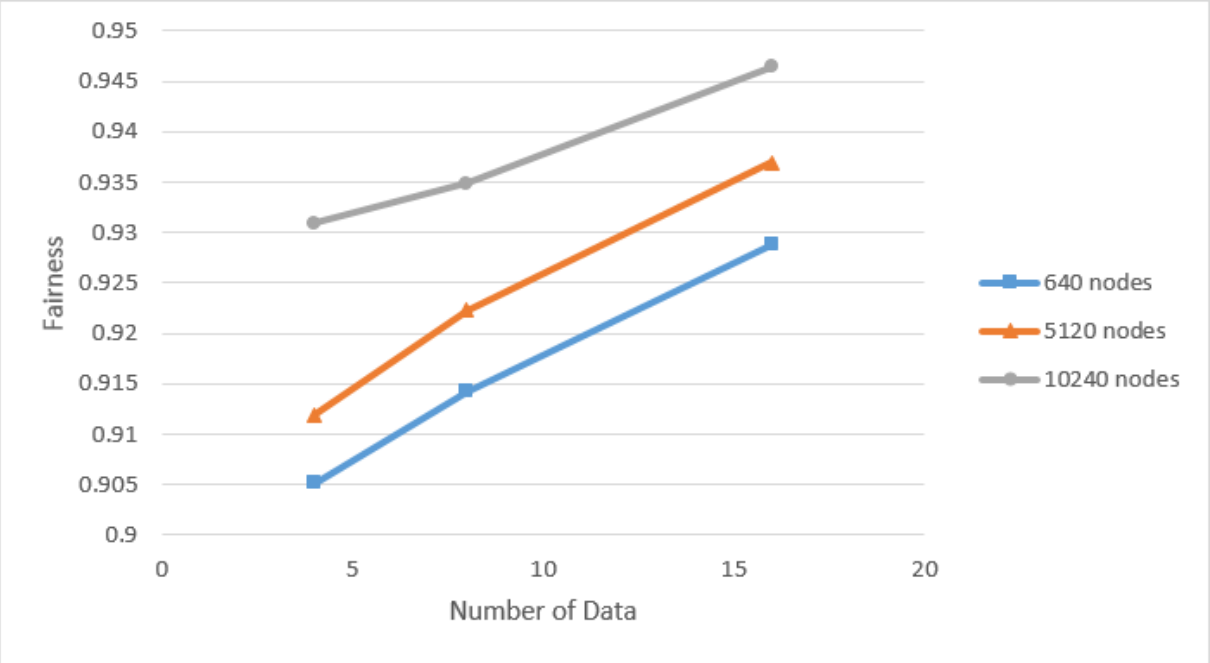


圖 27 總點數與資料總數對雜湊式複本位置函數負載平衡的影響

4.2.2 雜湊式複本位置函數與 Globus 複本位置函數的公平性比較

接下來我們把雜湊式複本位置服務與 Globus 複本位置服務做比較，圖 27、28、29 是在 r = 4、8、16 的時候，Globus 複本位置服務可以決定要把複本放在前點還是後點，而放的位置不同也會有不同的影響，在章節 4.1.2 可以得知全部點放在前點的跳躍數最少，但由圖 27、28、29 得知負載平衡就較不好，可以看到在總點數 640、1280、2560、5120、10240 的情況下，在犧牲一點跳躍數的情況下，雜湊式複本位置服務獲得最好的負載平衡。

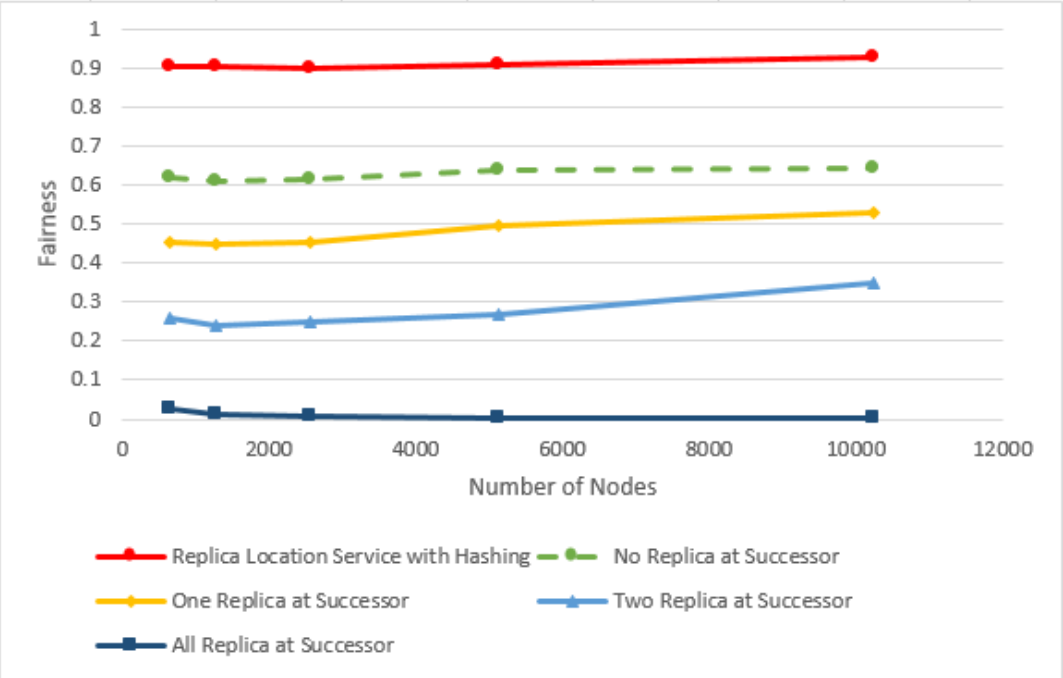


圖 28 r = 4 時，雜湊式複本位置服務與 Globus 複本位置服務下複本儲存在不同位置的公平性比較

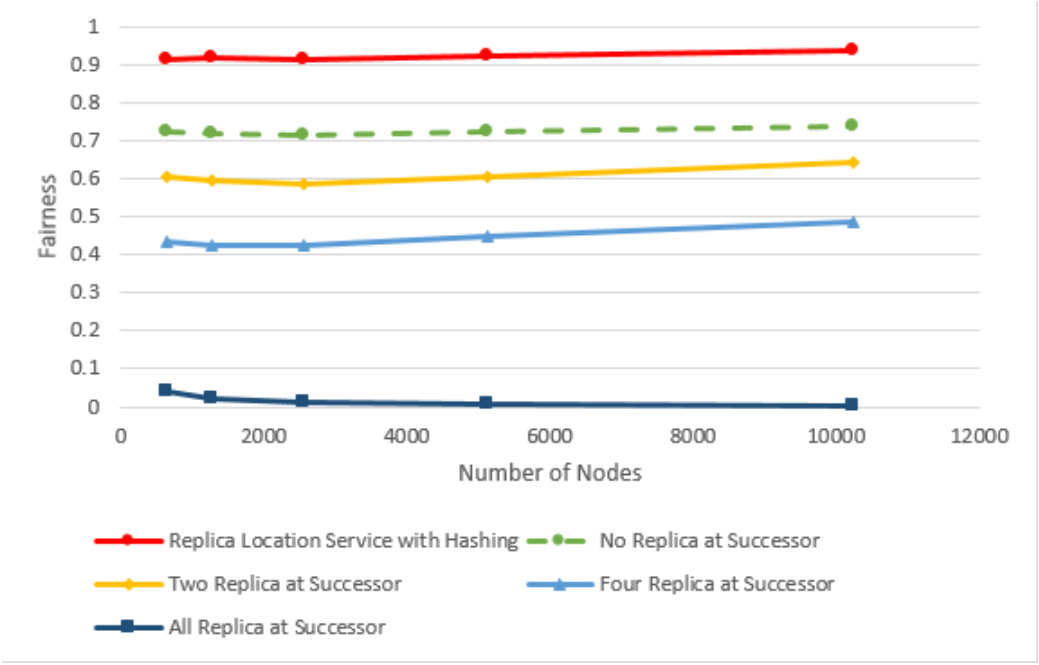


圖 29　*r* = 4 時，雜湊式複本位置服務與 Globus 複本位置服務下複本儲存在不同位置的公平性比較

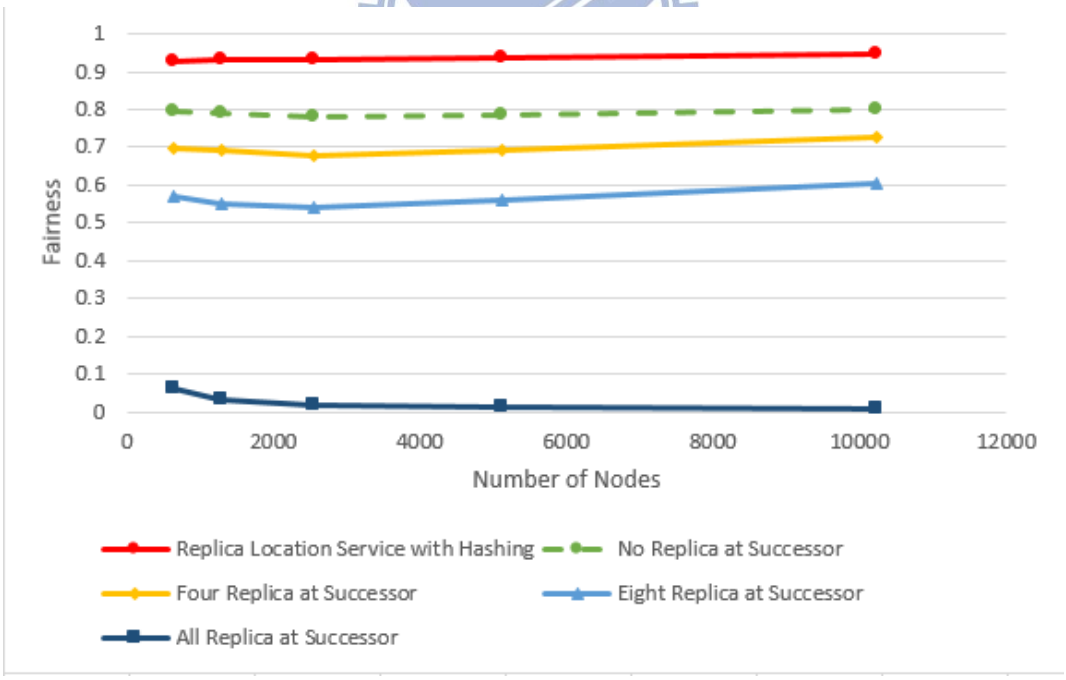


圖 30　*r* = 4 時，雜湊式複本位置服務與 Globus 複本位置服務下複本儲存在不同位置的公平性比較

4.3 模擬點的失效所造成的資料遺失

接下來我們模擬點失效的時候對 Globus 複本位置服務跟雜湊式複本位置服務的影響，情況是在總點數為 10240 的情況下，在每 10000 次資料搜尋要求會發生 10%、20%、30%、40%、50%的點數失效，然後去看他資料遺失的比例，當資料遺失率越低時，同時代表著資料的可靠性越高；我們拿出搜尋速度比雜湊式複本位置服務快的 Globus 複本位置服務來做比較，來顯示雖然雜湊式複本位置服務犧牲一點速度來換得良好的資料可靠性，如圖 30 為 $r = 4$ 的模擬，可以看出全部複本放在前點的 Globus 複本位置服務資料遺失率是高於雜湊式複本位置服務的，而隨著複本放越多在後點則資料遺失率會漸漸下降，在 30%以下的點數失效情況下，雜湊式複本位置服務的資料遺失率會比一個複本放在後點的 Globus 複本位置服務來的好，而超過 30%的點數失效的情況下會比較差的原因是因為失效率太高導致很容易超過雜湊式複本位置服務的搜尋生命周期，為了避免搜尋生命周期過長，通常會設置在理論的跳要長度(公式 3.3)，因此失效率太高時，便會有較高比例超過搜尋生命周期，導致資料遺失率增長，一般而言只要調高搜尋生命周期便可解決，只是會耗費更多跳躍數。

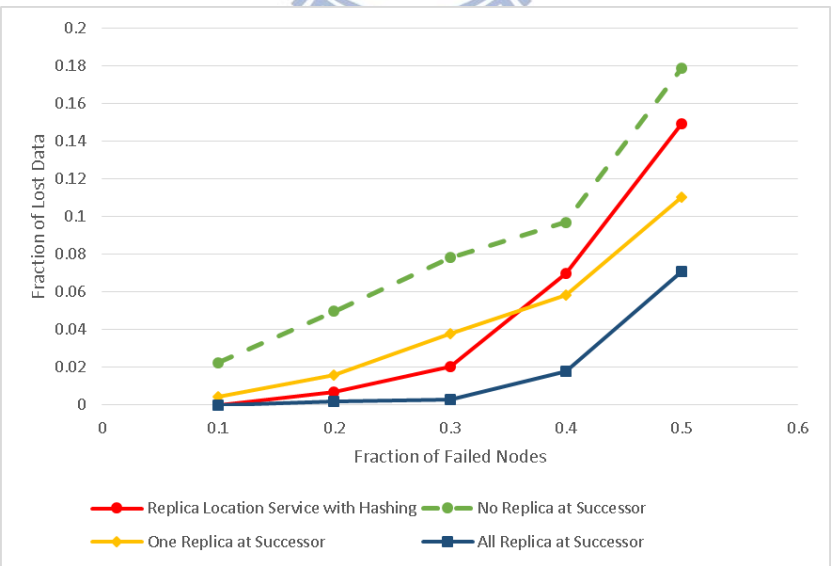


圖 31 r = 4 時，雜湊式複本位置服務與 Globus 複本位置服務下複本儲存在前點位置的資料遺失率比較

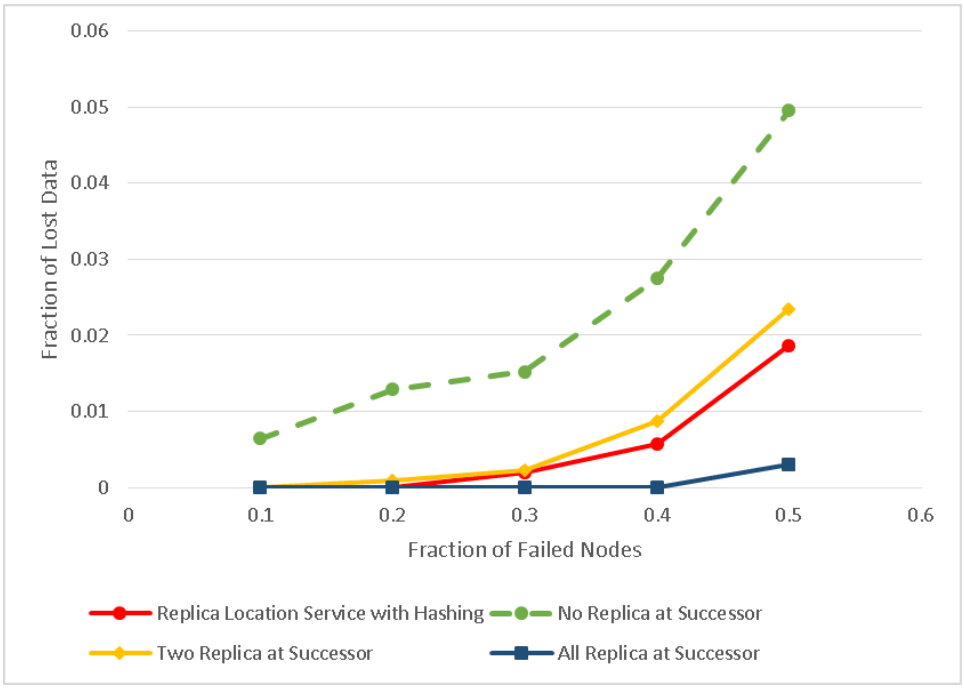


圖 32　r = 8 時，雜湊式複本位置服務與 Globus 複本位置服務下複本儲存在不同位置的資料遺失率比較

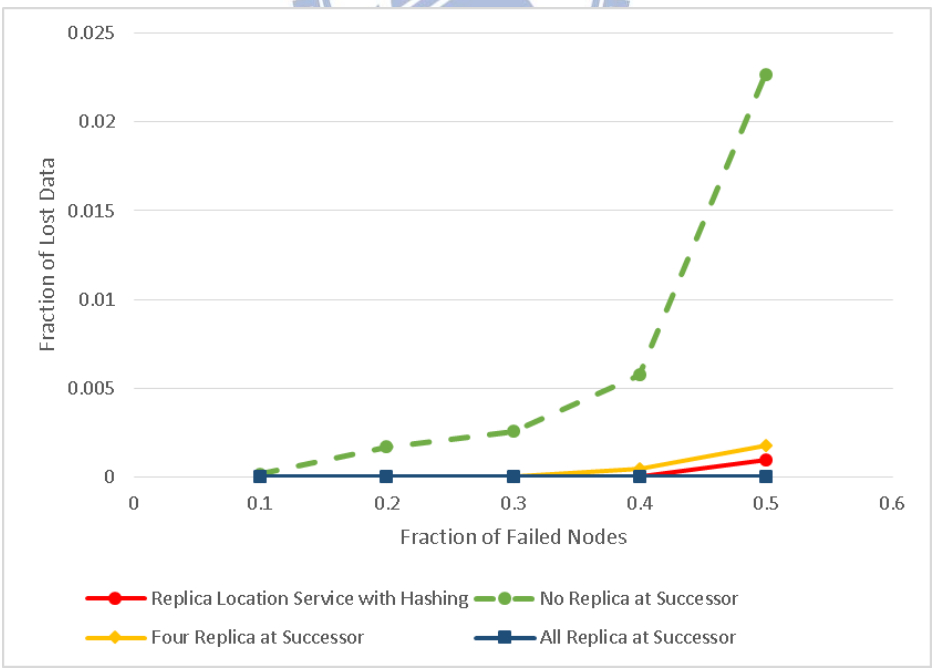


圖 33　r = 16 時，雜湊式複本位置服務與 Globus 複本位置服務下複本儲存在不同位置的資料遺失率比較

同理如圖 31、32 為 r = 8、16 的模擬，顯示出來雜湊式複本位置服務的資料遺失率都比較低，提供良好的資料可靠性；除此之外我們也可以看到，當複本數增加，資料可靠性也會上升，當 r= 16 時，即使將近 50%的點失效了還能維持 99.9%的資料可靠性；雖然無法達到把所有複本放在後點的 Globus 複本位置服務一樣的資料遺失率，但與之相比我們有最好的負載平衡跟較好的跳躍數。

4.4 維護複本所需的跳躍數

最後我們來討論維護複本所需的跳躍數，我們模擬了在不同複本、不同點數之下，各個情況所需的跳躍數，可以發跳躍數到達一定的點數之後，變增長緩慢，因此當點數越多時導致維護複本所需的資源並不會增加太快，符合在對等網路的情況，而且當複本數量越多時，跳躍數也會跟著下降，使維護複本速度更快。

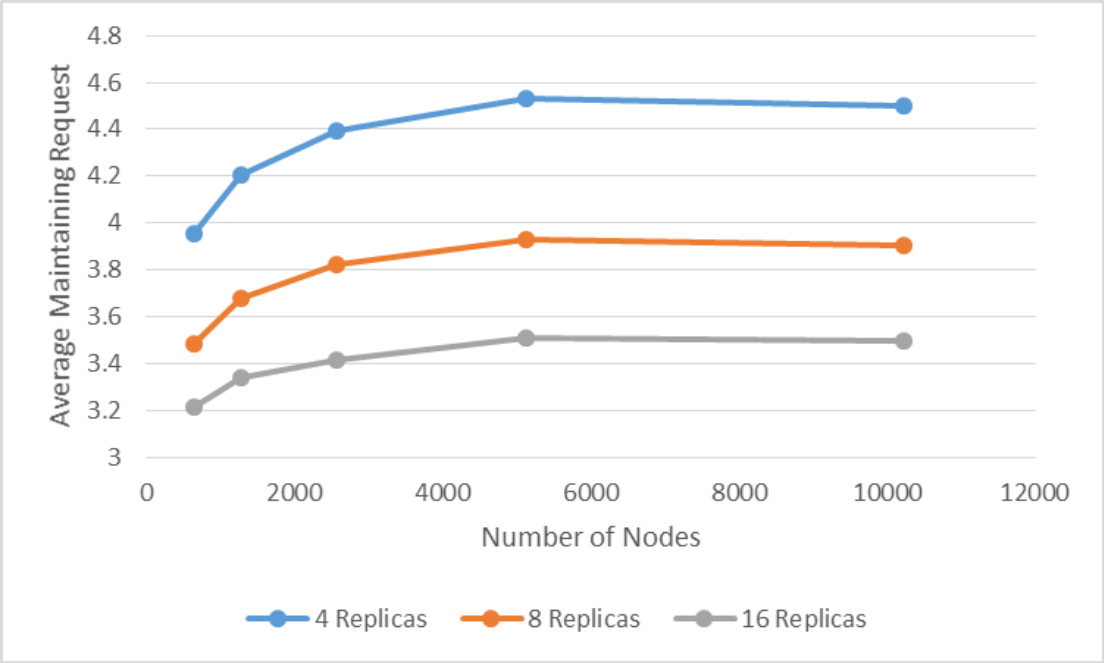


圖 34 維護複本所需的跳躍數

第五章

結論

很多分散式對等網路下的應用都需要利用到儲存的資源，Chord 協定提供一個很好的搜尋資料的分散式方法，本論文提供一個很好的方法：雜湊式複本位置服務，結合 Chord 協定與複製技術，並增加系統的健壯性與更好的負載平衡；

在雜湊式複本位置服務下，指向表只需儲存$O(logN)$其他點的資訊，便能使搜尋資料的跳躍數減少，平均為$\frac{1}{2}\ log_2(\frac{N}{r})$。

即使結合雜湊式複本位置服務，Chord 協定依然保有原本的特性：簡單、良好的效率、資料正確性，更加強了資料的負載平衡、更好的效率、自我恢復性、資料可靠性，最後的模擬顯示雜湊式複本位置服務在犧牲一點搜尋速度下，其本身的負載平衡與資料可靠性是非常好的。

在應用的層面上，雜湊式複本位置服務能使用在巨量點數的對等網路或分散式應用程式上，例如：分享共用檔案、分散式儲存系統、分散式目錄索引系統與分散式運算系統等，在這些系統上非常需要良好的負載平衡與系統健壯性來維持整個系統的服務，我們相信使用雜湊式複本位置服務的 Chord 協定，能滿足其要求。

參考文獻

[1] I. Stoica, R. Morris, D. Liben-Nowell, D.R. Karger, M.F. Kaashoek, F. Dabek, and H. Balakrishnan, 'Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications', IEEE/ACM Trans. Netw., 11 (2003), 17-32.

[2] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, 'A Scalable Content-Addressable Network,' Proc. ACM SIGCOMM, 2001.

[3] A. Rowstron, and P. Druschel, 'Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems', In Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001), 11 (2001), 329–350.

[4] B.Y. Zhao et al., 'Tapestry: A Resilient Global-Scale Overlay for Service Deployment,' IEEE J. Selected Areas in Comm.,vol. 22, 2004.

[5] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin, "Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web", in Proceedings of the twenty-ninth annual ACM symposium on Theory of computing (El Paso, Texas, United States: ACM, 1997), pp. 654-63.

[6] FIPS 180-1. Secure Hash Standard. U.S. Department of Com-merce/NIST, National Technical Information Service, Springfield, VA,Apr. 1995.

[7] A. Chervenak, and M. Cai, "Applying Peer-to-Peer Techniques to Grid Replica Location Services," Journal of Grid Computing, vol. 4, no. 1, pp. 49-69, 2006.

[8] http://www.globus.org/

[9] R. Jain, G. Babic, B. Nagendra, and C. Lam, “Fairness, call establishment latency and otherperformance metrics,” Tech. Rep. ATM Forum/96-1173, ATM Forum Document, August 1996.

[10] Y. Zhou; Koyanagi, K., "Improving routing load fairness in structured P2P overlay networks," Advanced Communication Technology (ICACT), 2013 15th International Conference on , pp.724,728, 27-30 Jan. 2013

[11] S. Kullback, "Letter to the Editor: The Kullback–Leibler distance". The American Statistician 41 (4), pp. 340–341, 1987

[12] Shin-Jung Yen, “An Information-theoretical Fairness metric for IEEE802.11 Wireless LAN,” Master thesis, National Sun Yat-sen University, 2004

101

碩士論文

應用雜湊函數在基於 Chord 協定的複本位置服務

交通大學

電機學院

電信工程研究所碩士班

李俊緯