南　華　大　學

資訊管理學系

碩士論文

無線廣播環境下關聯資料的快取配置方法

An efficient correlated data cache strategy for wireless server

研 究 生：童智揚

指導教授：蔡德謙　博士

中華民國　九十五年 六 月

南　華　大　學

資訊管理學系

碩 士 學 位 論 文

無線廣播環境下關聯資料的快取配置方法

研究生：童智揚

經考試合格特此證明

口試委員：吳光閔

_____

_____

_____

指導教授：蔡德謙

系主任(所長)：資訊管理學系 系主任 吳光閔

口試日期：中華民國 95 年 6 月 29 日

誌　　謝

這一頁是本篇論文最沒有壓力的一頁，經過了 2 年的努力終於能夠順利畢業，在天上的父親，感謝你默默的保護我，感謝母親和姐姐不斷給我鼓勵給我打氣，讓我能順利完成論文，謝謝佳晏，感謝 325 的無線夥伴士颺、昭勳、淑玲，還有坤錨，謝謝小吟的鼓勵(你不要再傻傻的笑了)，謝謝嘉明教我寫 java(每次都給你添麻煩真是抱歉)，還有乾訓學長、閔皓學長、民哲學長、俊杰學長、美倫學姊的幫忙，謝謝你們讓我能早日習慣研究所的生活，研一的學弟妹小賴、雅君、珊珊、岳勳、鎮宇，你們都是最好的。

感謝蔡德謙老師的細心指導讓我在研究所的 2 年獲益良多，謝謝老師不厭其煩的教導我程式，透過老師的教導讓我成長許多，希望將來能有成就以不負您的教誨，謝謝老師。

無線廣播環境下關聯資料的快取配置方法

學生：童智揚　　指導教授：蔡德謙 博士

南　華　大　學　資訊管理學系碩士班

摘　　要

在這個資訊大量充斥的時代中，對於資訊的需求成長仍不斷增加，而為因應的資訊需求的不斷成長，資訊傳遞媒介及其傳遞方法亦不斷演進。無線廣播即是因應資訊需求的成長及客戶端移動性變大的資訊傳播方法。當無線環境的使用者日益增加，為了讓大量的客戶端使用者能有效率、快速得到所需求的資料，廣播需求量較大的資料項是常被使用的方法之一。此種資料推播的方法能使客戶端預先快取暫時未用到的資料，而有效率的快取配置策略則能在有限的快取記憶體下快速提供客戶端查詢請求的資料項。

限於一般無線環境的頻寬均相當有限，我們這篇論文主要在於提出一個快取伺服器對相關聯廣播資料的快取配置方法。我們提出的最佳化快取配置方法考慮客戶端的查詢請求常包括一個以上的可重覆資料項，並藉由置換具有最小需求的資料項以降低客戶端所提出查詢請求後的平均等待時間。 最後經由實驗可證實我們的快取配置策略能夠比LRU快速

且有效率地降低客戶端的平均查詢讀取時間。

關鍵詞: 無線廣播、快取、關聯資料

An efficient correlated data cache strategy for wireless server

Student：Chih Yang Tung Advisors：Dr. Derchian Tsaih.

Department of Information Management
The M.B.A. Program
Nan-Hua University

ABSTRACT

Requirement of information keep growing up even in the time of information explosion. In response to this continuous requirement for information, number of data delivery medium and data spreading method are also increasing. The wireless broadcast is the new way of spreading information, which is also one of the answers to the growing demand of information and increasing mobility of client. Since the number of mobile application and mobile user keep increasing nowadays. Broadcasting the hot data for clients' query request is one of frequently used method which can efficiently convey data to vast number of clients. Mobile host can constantly read from the broadcasted channel and cache the hat data into memory in order to have a quick response for frequently submitted query.

Due to the bandwidth limitation in wireless environment, we propose a cache strategy for broadcasted data in cache server which consider the situation that different clients' query request can have same data items, by replacing the data item with minimum demand, the average clients' query request can be minimize. Via the result of simulation experiments, by using our proposed cache strategy, the response time for average clients' query request can be reduced.

Keyword: wireless broadcast,Cache,correlated data

目 錄

表　目　錄

圖　目　錄

第一章　緒論

近年來由於電腦硬體的發展不斷往無線通訊（Wireless Communication）領域靠攏，使得越來越多的行動設備（Mobile Devices）能在無線行動通訊網路下使用行動服務（Mobile Services），例如PDA或筆記型電腦等，藉此在無線行動通訊網路環境下的使用者（Mobile Users）能隨時隨地使用此資訊服務，但網路在大量使用者的要求出現時，伺服器為了要處理大量的要求而負載過重，導致伺服器無法處理目前的所有要求。然而廣播系統架構、網路資訊傳遞、行動計算、廣播系統的使用，使得一台擁有大量有用資料的資訊中心或是分散式資料庫，能服務滿足非常大量的用戶的要求，這些用戶端能經由廣播系統有效率地得到這些他們所需求的資料。

第一節　研究背景與動機

寬頻網路的產生，使得人們在搜尋資料、觀看網路電視以及線上收聽音樂都更為方便、迅速，免去網路壅塞問題。而現今無線網路的萌生，更可以讓人們透過無線網路的連結，隨時隨地就能夠使用PDA或者筆記型電腦搜尋所想要的資料，再也不用擔心環境與設備因素的限制。

無線環境的頻寬限制相當大，它無法擁有如同有線網路一般大的頻寬；無線網路也可能因為地形或者訊號的強弱問題而發生延遲或中斷的

狀況，因此，網路頻寬成為無線環境中最需要注意以及吸引最多人所研究的議題。目前大多數的研究都是著重於要如何能讓伺服器更有效率的去發佈及傳遞資料，如此一來就會牽涉到如何統計哪些熱門的資訊較常被使用者需要，並以此去分配所需的資料，用以節省有限的頻寬，也以最少的時間提供服務給有此需求的使用者。而對此研究我們除了探討資料的被讀取率，還考慮了資料項與資料項之間的關聯性，在無線廣播環境裡網路頻寬是有限的，但資料是無限的，因此，我們可以藉由探討資料項與資料項之間的關聯性來有效地降低使用者的等待時間。而如何能藉由資料項與資料項之間的關聯性來降低使用者的查詢讀取時間正是本篇論文的研究主題。

第二節　研究主題

在無線網路的快取伺服器環境裡有許多廣播的種類，但是主要分為客戶端快取(Client Cache)、代理快取(Proxy Cache)與主機快取(Server Cache)三種模式，不同的快取模式其運作方式與傳送的資料方法也不一樣。本論文將專注於研究快取伺服器裡廣播資料的排程議題，之前有許多的文獻在研究、探討快取伺服器的廣播資料排程議題[2][3][5][6][7][8][9][10] [11][13][14][17] [18]，許多研究者針對不同的環境問題提出不同的解決方法，目的都在於希望能有效降低使用者的等待時

間，但多數的研究都只是單方面的探討資料的排程與資料項的存放位置，鮮少有研究者注意到資料項與資料項之間的關聯性。

因此我們將以研究資料項與資料項之間的關聯性作為本篇論文的研究主題，期望我們所發展的演算法能夠在隨機產生資料的環境中有效地降低使用者的等待時間。

第三節　研究限制

我們的研究無法應用於各種無線網路環境，因此將其限制說明如下：

1.本研究方法應用於無線廣播中的快取伺服器。

2.本研究的最小需求演算法適用於單頻道廣播系統，在本研究的議題裡面暫不考慮多頻道的廣播系統。

3.本研究的方法能有效改善客戶端使用 LRU 的查詢讀取時間。

第四節　簡介最小需求置換演算法

本篇論文首先介紹無線網路的整體環境，以了解無線網路的運作模式，其次描述我們提出的最小需求置換演算法。

從目前現有的相關文獻裡可以發現，多數的研究主要在於單一資料的排程方式與等待時間的相關議題。而單一資料的排程問題主要在於資料

項的被讀取率在與客戶端對Server查詢資料項而有所改變，在這種情形之下，只要每次使用者提出請求時，Server端就必須重新計算資料的被讀取率，這對Server端的負擔非常大。等待時間的問題在於被讀取的資料可以區分為所謂的熱門資料與冷門資料，所謂的熱門資料，即為常常被使用者請求的資料，由於讀取率高，相對的等待時間則較短；相反的，冷門資料被請求的次數過少，但是這不代表此資料項是不需要的，所以使用者在請求冷門資料時往往需要長時間的等待，若錯過等待時間就得花上較長時間請求資料。

針對上述的兩個問題以及應用於無線網路的環境下，我們提出最小需求置換演算法(MinDemand，MD)，本方法應用資料項與資料項之間的關聯性來運作。MD的方法運用於快取伺服器的網路架構下，能有效提高並穩定Server的運作，並且確實有效降低客戶端的平均查詢讀取時間。

MD主要運作方式在於利用搜尋的方式，如此一來可以去除煩雜的運算方式，有別於以往的方法在接收到廣播值之後開始往後做搜尋的動作，搜尋可以被替代的資料項。最小需求置換演算法(MD)是在接收到廣播值之後，從廣播值開始往前搜尋可以被替代的資料項，假使接收到的資料項有儲存在快取伺服器，則更新此資料項的被讀取時間；若此資料項沒有儲存於快取伺服器，則往前尋找下一個資料項，查看此資料項被

哪些query請求，比較query請求資料項的關聯性，找出有儲存在快取伺服器裡面，而且最小需求值最大的資料項置換成目前的廣播值。由於考量等待的時間問題，往往為了等待某一資料，需要等待一段時間，造成電力浪費或者Server的負載過大，在MD的方法裡，為了讓使用者降低等待時間，本研究考量資料項之間的關聯性，透過資料項之間的關聯性把暫時不需要的資料項或尚未讀取的資料項置換成目前的廣播資料項，以降低使用者的等待時間。

因為MD也把冷門資料納入考量的範圍，所以在找尋資料項時同時，亦加入資料的被請求頻率，以便冷門資料能有較多的讀取機會，藉以降低使用者在讀取冷門資料時的等待時間。

在模擬結果顯示，我們的方法在隨機產生資料項的環境中，能比 LRU 有效降低使用者的等待時間，最小需求演算法可以快速地藉由資料項與資料項之間的關聯性，找尋在快取伺服器裡可以置換的資料項，有效地提升 Server 端運作，並降低使用者的等待時間。另外，模擬結果也表現出在資料項目數量為 100，query 為 200，當快取伺服器大小為 20、30、40，偏斜度在 0.2、0.4、0.6、0.8 時，最小需求置換演算法可以比 LRU[2] 有效降低客戶端的查詢讀取時間。

本篇論文其餘的部份將有組織的簡述如下：第二章將討論在無線網

路環境下的相關議題的探討；第三章則是描述透過文獻探討之後，所發現的問題；接著，第四章詳細介紹說明我們演算法的概念、程序及運作；我們將模擬實驗的結果介紹於第五章，此章節將詳細說明模擬環境、參數及結果數據分析；最後第六章是本研究的結論。

第二章　文獻探討與相關研究

第一節　廣播環境

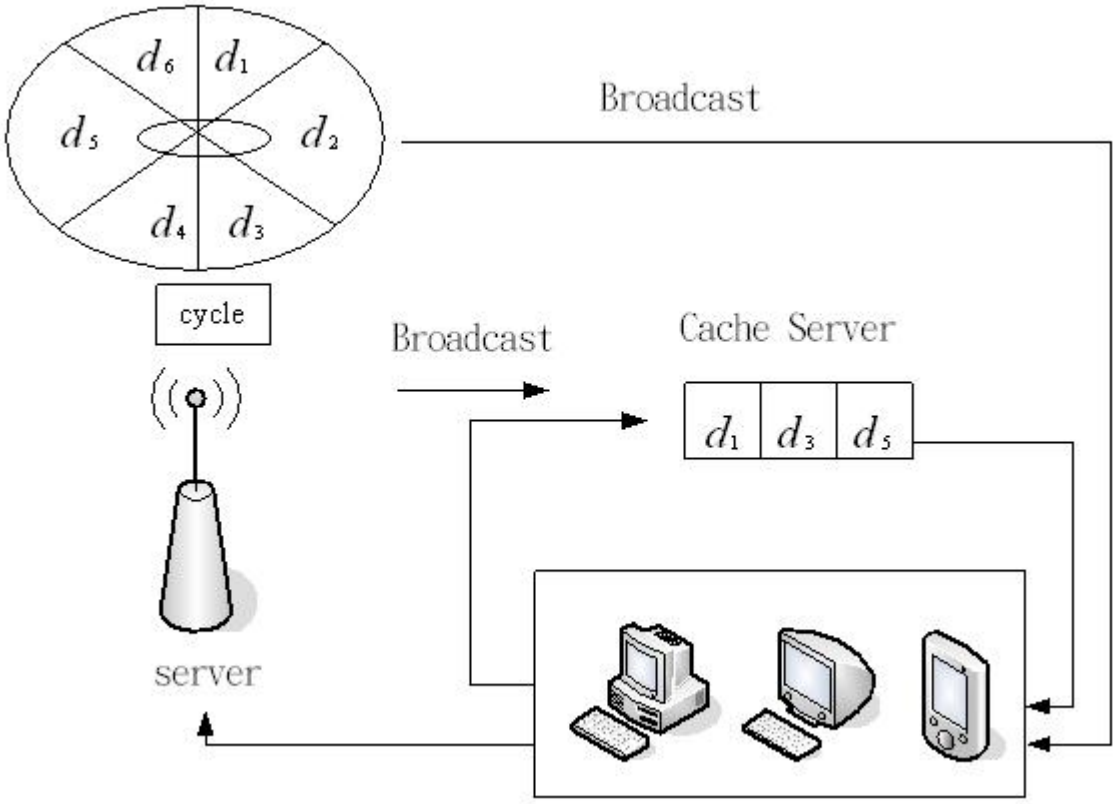


圖 1 無線網路廣播環境概念圖

日常生活中，以廣播方式傳播資訊的事務日漸增加，以往在網路上以單點傳送機制來滿足使用者提出的資料項要求，有鑑於多數人會要求相同熱門的資料項，提供資料項的 server 卻不停發出相同的資料項給各個使用者，當 server 遇到大量要求的時候，server 會不斷傳送相同的資料項

給要求者，導致服務的效能降低。因此，廣播快取伺服器廣泛被使用，儲存大眾常請求的資料項在快取伺服器上，讓大多數的使用者能利用監聽(monitor)的方式得到自己想要的資料項，大大的減少 server 重複傳送相同資料給各個要求者的動作，並將這種有效率的傳送運用在一般生活中，例如線上觀看網路新聞、線上收聽廣播電台、股票市場的買賣交易資訊等等，都能有效的將熱門資訊放在廣播媒介上，讓大量的需求者能透過這個媒介得到想要的資訊。

一個無線廣播環境包含了大量的客戶端(client)以及用來存放在用戶端提出要求的序列(request queue)與一個在後端所選定的排程演算法(Scheduler)的伺服器(broadcast server)和負責擔任廣播媒介的廣播頻道。圖一顯示許多用戶端監聽循環撥放的資料並等待自己想要的資料。

第二節　無線網路的討論議題

WLAN的標準於1997 年8 月由電子電機工程協會(IEEE, Institute of Electrical and Electronic Engineers)制訂、公佈，並制定802.11 的正式標準，目前市面上的無線網路 802.11g 是屬於802.11 標準的另一個延伸，其規格的產品從2000 年第四季開始已經大量地普及在商業的應用上，佔有率在現階段是最高的。一些廠商會將無線網路設備產品的傳輸速度和傳輸距離通稱為「涵蓋範圍」，通常以54Mbps/50公尺這樣的方式表示無

線網路設備的傳輸距離和速度。無線網路設備會隨時偵測無線電波的強弱，然後依照環境不同自動調整傳輸速率，因此傳輸距離愈長，速度愈慢。在802.11g 規格中，傳輸速度最快為54Mbps，但在實際上並沒有如此完美。無線網路連線品質不穩定易造成斷線，如無線基地台佈置量太少，導致涵蓋範圍太小，造成許多的收訊死角。無線網路傳輸品質隨著上網環境的不同而改變，例如：距離遠近、有無障礙物的阻擋以及其他的無線電波干擾等都是影響速度的重要關鍵，尤其是一般家庭或大樓的樑柱都會大幅影響無線網路的傳輸品質，因此在無線網路的涵蓋範圍內愈空曠愈好。一般正常情況下，無線網路傳輸速度在200kb/ s~600kb/ s。

根據無線網路的發展，我們發現無線技術仍受限於網路頻寬、傳播媒介等因素。因此在廣播的環境中，所面臨有待克服的議題還有許多，最為重要的話題是「平均等待時間的過長」、「無法即時的處理大量的資料項請求」，多數研究的探討主題著重在如何有效地縮短客戶的平均等待時間[4]。平均等待時間的降低，直覺上是將熱門的資料做多次的廣播，讓絕大部分的使用者能很快的滿足，但是，卻使一些只要求冷門資料的使用者被忽略了，形成所謂挨餓狀態(Starvation)。因此也有部分的研究加入了資料關聯性的考量，只要使用者有要求及廣播系統有此資料，一定能在理想的時間內滿足使用者的需求。

第三節　何謂快取

快取一種特殊的高速儲存機制，其可為主記憶體的保留區或獨立的高速儲存設備。快取記憶體是一小塊反應相當快的記憶體，它是為了加速內部資料和軟體指令傳輸的特殊目的而設計的。但快取記憶體不夠大到用來儲存所有的資料，因此並不是所有想要的資料都能在快取記憶體中找到。不過，越常使用的資料就越有機會被放在快取記憶體裡，而最後所得的結果就是增進了處理的速度。PC 通常用到兩類快取(cache)：memory cache and disk cache，memory cache 有時稱為 cache store 或 RAM cache，其採用高速 static RAM(SRAM)而不用主記憶體所用較慢及較便宜的 dynamic RAM(DRAM)。memory cache 是有效的，因為大部分的程式會重複的存取相同的資料或指令，盡可能將這些資訊存在 SRAM，電腦可避免存取較慢的 DRAM。

某些 memory cache 被加入微處理器的架構，例如 Intel 80486 微處理器包含 8KB memory cache 而 Pentium 有 16KB memory cache，此種內部 cache 通常稱為 Level 1(L1)cache，大部分現代 PC 也有外部 cache memory，稱為 Level 2(L2)cache，這些 cache 位於 CPU 和 DRAM 間，如同 L1 caches，L2 caches 由 SRAM 組成但 L2 大的多

disk cache 和 memory cache 以同樣的原則運作但非採用高速

SRAM，disk cache 用傳統的主記憶體，最近一次由磁碟存取的資料存在記憶體暫存區，當程式需要從磁碟存取資料，它先檢查磁碟 cache 看看資料是否在 cache，disk cache 可大大的增加應用程式的效能，因為存取 RAM 裡面 1 byte 資料的速度比存取硬碟 1 byte 資料快數千倍。

當在快取中找到資料，稱為 cache hit，而快取的有效性由它資料的命中率判定，很多快取系統用稱為 smart cache 的技術，此類系統可識別某些類型的常用資料，決定那些資訊應該存在快取的策略構成快取置換的議題。快取記憶體(Cache RAM) 是一個電腦內置的臨時儲藏地方專為經常或最近被獲取的資料而設。在電腦內有不同種類的快取記憶體。 例如，您的瀏覽器存放了最近被參觀的網頁在快取記憶體目錄在內，以便您不需向原始伺服器提出要求的指令便能瀏覽網頁。當您點擊中"更新"按鈕，被您的瀏覽器所儲藏的網頁會與前時期在網路所存取的做比較，如果需要的話就會更新您的地方版本。您可自設您的瀏覽器的快取記憶體和磁盤高速緩存的大小優化您的上網速度。另外二種個人電腦所使用快取記憶體。快取記憶體(或高速緩衝記憶體) 是一部份電腦存取空間包括高速 RAM (隨機存取存儲裝置) ，它們是特別設計去配合快速資料檢索。最後就是磁盤高速緩存，在我們的硬碟存放最近被獲取的資料，改善效能和速度。如讀取或寫入資料時，將資料暫存快取記憶體 (Cache

Memory)中，將會加快運算速度的一種處理模式。以修車的工作舉例來說，技工從工具室，用工具箱裝來一些他認為會用的到的工具，一旦開始作業後，他可能發現上次所拿的工具不對或不敷使用，則他必須拿工具箱回到工作室去再取一次；再作業一段時間後，他可能發現上次所拿的工具又不方便使用了，則他必須回到工作室去再取一次。如此反覆，直到他將修車的工作完成。如果他的工具箱很小，代表他一次所能裝的工具很少，則他來回的次數就多，耗費的時間及能量就多；相對而言，如果他的工具箱較大，代表他一次所能裝的工具較多，則他來回的次數就少，耗費的時間及能量就少，整體工作效率便會增加。 在此，「工具箱」代表「快取記憶體」，「工具室」代表「硬碟」，「工具」代表「資料」。使用應用程式時，電腦會將一些資料〔工具〕，暫存於快取記憶體〔工具箱〕中，避免重複至硬碟〔工具室〕存取，以節省時間及精力，加快工作效率。快取記憶體〔工具箱〕愈大，來回的次數就愈少。如奔騰四代〔Pentium 4〕使用 512KB 快取記憶體，比一般的 128KB 或 256KB 多上二至四倍，則電腦處理資料的速度自然加快許多。但注意的是，電腦處理資料的速度，主要還是取決於中央處理器〔CPU〕的快慢，快取記憶體的容量只是作為一種輔助，還要有其他硬體、軟體及各項條件的配合，才能發揮整體工作效率。一般而言，當我們要從儲存設備中存取

資料時，都是透過檔案系統的操作來完成。這些資料檔案都是以「資料區塊 (block)」為單位保存在檔案系統中，而系統在存取時也是以整個資料區塊為單位來讀寫。資料區塊的大小依作業系統、機器硬體結構、以及儲設備容量大小等不同而有不同，以 GNU/Linux 系統為例，常見的資料區塊大小為 1024、2048、或 4096 個位元組。在儲存設備與檔案資料讀取介面之間，還有一層稱之為「資料區塊快取層」，由檔案系統直接管理，其主要的功能是做儲存設備上的資料區塊快取。當系統要讀取某些區塊時，檔案系統在真正去讀取儲存設備之前，會先檢視快取層的資料，如果找到所需的區塊就直接傳回，若找不到時檔案系統才會真正去讀取儲存設備，並將讀取的結果暫時保存在快取層中。同理，如果系統要寫入某些資料區塊時，檔案系統也會先將要寫入的內容保存在快取層中，等系統較不忙碌或儲存設備沒在使用時才做真正的寫入動作。因此，透過快取層的運作，可以減少直接由儲存設備存取資料的機會，因而大幅提升系統的讀寫效率。

然而快取層的優點同時也帶來了負面效應。由於新寫入的資料不是立即寫入儲存設備中，故如果在此過程中突然發生了硬體錯誤或斷電，將會造成資料的流失。同時，在某些系統中，其快取層的運作效率也可能不盡理想。因此，在某些特殊的應用場合中，如保存極重要的資料，

或大型資料庫系統的管理運作等，我們需要直接讀寫儲存設備，而不需要經由檔案系統為媒介。建置快取伺服器(Cache Server)的主要目的在於儲存常被存取的熱門資料，讓使用者可以在鄰近網路內就近取得所要的資料，而不必每次經由 WAN 向原始網站存取。此應用提供增加存取速度、降低網路流量和管理使用者存取網路資源等優點。

第四節　快取機制相關探討

網路上的技術發展也讓使用者可以透過 World Wide Web[12]取得更多的資訊，使得網路的人口快速增加，資料項大增，網路壅塞的情形相對的也越來越嚴重，如何能有效率的提升存取的效能，降低使用者的等待時間將是重要的議題。

表 1 快取伺服器的分類

	快取位置	快取作用
Client Cache	客戶端的瀏覽器，包含本身的記憶體以及可永久儲存的磁碟空間。	減少使用者因為等待資料傳輸而產生較長的等待時間。
Proxy Cache	主要分為兩種，第一	減少因為傳送資料而產生

	種是與客戶端直接連接的區域代理伺服器，第二種是位於由網路上或者中樞網路的代理伺服器。	較長的等待時間，避免相同的資料重複傳送，而造成網路頻寬的壅塞與浪費，可以降低熱門網頁伺服器的負載。
Server Cache	原始的網頁伺服器。	降低網頁伺服器的負載。

資料來源:陳佳慧(民國 89 年)

改善效能的方法可以採用叢集的架構平行處理使用者的請求，或者是提高網路頻寬，然而相對於資料量和使用人口的成長來說，真正能有效的改善方式是利用快取的方式，使用者透過快取來取得資料，不僅可以減少因為網路上的相同資料傳輸，所造成的頻寬浪費，也可以讓使用者能找尋最近的伺服器以方便取得資料，減少使用者的等待時間。

這些用來改善效能的快取，依據其所扮演的角色我們可以分為三種，如同表 1 所表示的，其中 client cache 雖然可以有效地減少使用者的等待時間，但是因為該快取位於客戶端，快取中的物件不能與其鄰近節點的客戶端共同使用，所產生的效能也僅僅由本身所獲得。而 Proxy cache [15][16]中的物件不僅可以提供給客戶端的瀏覽器使用，也可以提供給其他鄰近節點的代理伺服器使用，這些代理伺服器可以架構成多層的組織

結構，不僅可以讓區域性的使用者減少因為檔案傳輸而造成的延遲時間，也可以減少網頁伺服器的負載，亦可避免在相同的網路中傳送重複的資料，所以目前的研究也都集中在 Web Poxy Server 的議題上。

為了有效率的提升效能，有許多的策略和架構在探討，大致可以分為下列幾項:

1.設計新的快取架構

一個好的快取架構可以有效的提昇web proxy cache的效能，藉由快取的合作機制，從其他鄰近節點的快取伺服器，就可以取得所需要的物件。例如使用virtual proxy graph [VPG]，透過proxy來連結幫助找尋cached web documents 和平衡工作量。再利用Cache Line[16]來做快取的置換策略動作。

2. 研究發展新的快取置換策略

快取伺服器用以存取物件的空間是有限制的，隨時都會有新的物件要進入快取伺服器裡面，而快取伺服器的空間又有限時，這時候就必須選擇一個或者是多個物件來做置換的動作，目的就是希望能有一個足夠的空間來置放新的物件。置換的策略在代理伺服器裡面所扮演的角色在於使得網路資源能被有效的利用，包括本身伺服器上的磁碟、記憶體空間和網路頻寬。置換策略可以說是快取伺服器的重要議題，所以也有許

多相關的議題是在探討快取伺服器的置換策略。

第五節　快取機制相關的資料置換方法

我們在提出部分置換策略同時，希望能達到下列目標：

1.Low Latency Time

Low latency time 代表的是客戶端的查詢讀取時間，在 Wooster[14]的研究中指出，高頻率的置換方法不代表能有效降低客戶端平均查詢讀取時間，主要的原因在於置換策略的方法不當，置換了需求頻率較高的資料項，導致使用者在等待回應的時間上並沒有獲得有效的改善，所以我們在提高置換率的同時也必須要求較低的查詢讀取時間。

2.High Hit Rate

Hit rate 代表的是快取中資料項的重覆使用率，hit rate 高的置換策略使得使用者在代理伺服器上即可取得所需要的資料，攔截大部分向原始 server 索取資料的請求，有效的降低熱門伺服器的負載。

2.5.1　Prefetch演算法研究

預測演算法[prefetch]，在預測使用者行為將進行之前，在網路負載較輕時，將所需的資料物件提早置放到快取伺服器裡面，以提高系統的效能。這個預測的動作可以客戶端或者伺服器端來進行，在預測準確的情況下，將可以有效的減低使用者的等待時間。以下為三種常被論及所預測的方式：

(1) Proxy side prefetching

此種方式適用於proxy 在使用者提出要求前，就預先將其所需的資訊取回到所連接的proxy，如此可減少傳輸延遲時間，降低重複的資料傳輸，避免網路頻寬的浪費，減低熱門網頁的網頁伺服器的負擔。

(2) Client side prefetching

此種方式是在使用者提出要求前，就預先將其所需的資訊取回client端，但是客戶端所預先取回的物件並不能讓其它使用者端來共享，所以使用者端的預載動作對於整體效能的改善程度相當有限。但是假如所需的物件有存在本機的話，由於不須再到網路上下載，因此可大大的改善反應時間。

(3) Server side prefetching

此種方式主要在server 端將所預測到的物件從硬碟移到主記憶體，假如有hit 到資料項則可改善server 的反應時間。

透過上述三個研究方向，都能有效的提高代理伺服器的效能，但對於開發新的快取架構而言，雖然能夠有效率的提高資料 hit rate 以及減少延遲時間，但是在開發新的架構時必須要考慮到其適用性及相關的網路架構，可以說相當複雜。

Pre-fetch 的效能好壞在於其本身的預測準確度[11]，預測的方法藉由長期觀察使用者的運作模式，預測使用者可能將會存取的物件，進而進行預測的動作。雖然高度的準確預測可大大的降低使用者的等待時間，

相反的，預測失敗不單單只是網路頻寬的浪費，更是佔據快取伺服器的空間，因為網路上的使用者其使用網路方式有所不同，所以 pre-fetch 無法有效的應用，因為到目前為止的研究都表示出，pre-fetch 的方法無法有效的帶來預期的效果，反而浪費有限的網路頻寬資源。

2.5.2 LRU

LRU 為Least Recently Used 的簡稱[2]，其主要作法為紀錄物件停留於快取記憶體內的時間，當有新的資料須置換於快取伺服器中，LRU 會將快取伺服器中最不常被讀取的資料項丟棄而以讓新的資料項取代，因為此方法只有考慮時間因素，並未參考其它的特性所以容易實作，但是 LRU 這種以物件最近一次被存取的時間作為評估置換與否的作法並非是最理想的，因為在每一個資料項被存取頻率相當時，LRU的方法並不是最佳的，反而會造成系統資源浪費。如圖2，假如目前正在廣播資料項為 d_4，因為存於快取中的資料項 d_3 timer值最小，所我們把資料項 d_4 置換於快取中 d_3 的位置上。

快取資料	0	1	2	3
Timer	110	257	758	12

置換前的快取伺服器

快取資料	0	1	2	4
Timer	110	257	758	908

置換後的快取伺服器

圖 2 LRU 的置換策略

2.5.3 LFU

LFU[3][7]為Least Frequently Used 的縮寫，其主要作法為紀錄快取記憶體中物件被重複存取的次數，然後選擇存取次數最低的物件優先置換，這個方法可能造成一個很嚴重的問題，就是某些物件因為過去一段時間的存取較為頻繁，存取次數達到了某一個預定值，使得即使將來再也用不到了，卻仍然會一直被保留在快取記憶體中而無法置換。

LFU 置換策略會率先選擇被使用頻率最低的資料項來置換，這樣的置換策略將會在快取中保留著被使用頻率較高的資料項，因此在考量資料項的需求頻率上會有較佳的表現。但是在網際網路上，資料的更新率很高，單純以LFU 作為置換原則，將會保留曾經熱門一時而存取次數相當多，現在卻無人存取的資料項，導致這些資料項在快取中佔據過多空間，降低了新進物件的保存率。如圖3，假如目前正在廣播資料項為d_4，資料項d_1被重複存取的次數是最少的，所以我們把資料項d_4置換於d_1的位置上。

快取資料	0	1	2	3
times	50	27	78	99

置換前的快取伺服器

快取資料	0	4	2	3
times	50	0	78	99

置換後的快取伺服器

圖 3 LFU 的置換策略

2.5.4 Size Policy

SIZE Policy [3]置換策略如其名所示，是以文件的大小做為置換的依據。當快取需要做文件置換時，SIZE 策略會將快取中最大的文件置換掉，以空出空間來容納更多小的文件。如此一來，快取的hit rate 便可以提高。然而由於SIZE 策略只考慮資料項的大小，使得SIZE 策略具有兩項缺點：(1)在快取中的文件大多是容量小的文件，雖然cache hit rate 高，但在byte hit rate 方面的表現卻未見理想。(2)給予小的文件較高的優先權可以留在快取中，造成其即使長時間沒有被存取，卻還是沒有被淘汰換掉。如圖4，假如目前正在廣播資料項d_4，資料項d_0的儲存byte是最大的，所我們把資料項d_4置換於快取中d_1的位置上。

快取資料	0	1	2	3
byte	111	30	57	87

置換前的快取伺服器

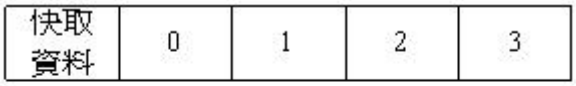
快取資料	4	1	2	3
byte	75	30	57	87

置換後的快取伺服器

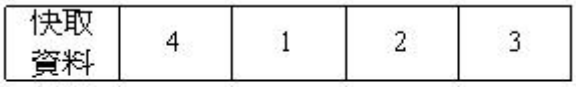
圖 4 Size Policy 的置換策略

2.5.5 CF

CF[4]為 Closest-First 的縮寫，快取的置換策略裡，CF 在快取伺服器裡會搜尋需要置換的資料項，假設目前接收到的廣播資料項已存在快取伺服器中，則更新此資料項的存取記錄時間；若沒有存於快取伺服器中，則往後尋找即將廣播但已存於快取伺服器中的資料項，將正在廣播資料項置換於此資料項的位置上。舉例來說，圖 5 假如目前正在廣播資料項 d_4，找尋廣播序列中下一個有儲存在快取的資料項 d_0，所以我們把資料項 d_4 置換於 d_0 的位置上。



置換前的快取伺服器



置換後的快取伺服器

圖 5 CF 的置換策略

2.5.6 SF

SF 為 Search Further 的縮寫，SF 為 CF[4]的改進，在快取的置換策略裡，CF 的置換策略是在快取伺服器中往後尋找即將廣播但已存於快取伺服器中的資料項，將正在廣播資料項置換於此資料項的位置上，但將資料項置換於此或許不能達到最低查詢等待時間，因此延伸 SF 的演算法。SF 的置換策略是在快取伺服器中往後尋找即將廣播但已存於快取伺服器中的資料項，找到置換此資料項能最有效降低客戶端的查詢讀取時間，並且將正在廣播資料項置換於快取中此資料項的位置上

2.5.7 CBS

之前的方法都是在做資料搜尋的動作，但是並沒有探討到資料的關聯性，直到 CBS[8]，此方法是往後作尋找資料的關聯性。CBS(Correlation Based Scheduling)的方法，在資料項中考慮到相互關係，為了簡化目的，我們假設客戶端在同一時間提出兩個資料項的請求。

CBS 的置換策略

1.計算每一個資料項與資料項之間的權重值，由節點組成表示全部資料項和所在位置連結成完整的圖，權重值W_{ij}為在資料項 i 和 j 之間的距離，$W_{ij} = 1 - P_{ij}$。

2. P_{ij}表示客戶端讀取資料項 i 與下一個讀取資料項 j 的機率。

3.Hamiltonian circuit選擇一條權重值最小的值當作是路徑，用來解決 traveling salesman problem (TSP)的圖。

4.沿著選擇的路徑，全部的資料項被放進廣播循環裡。

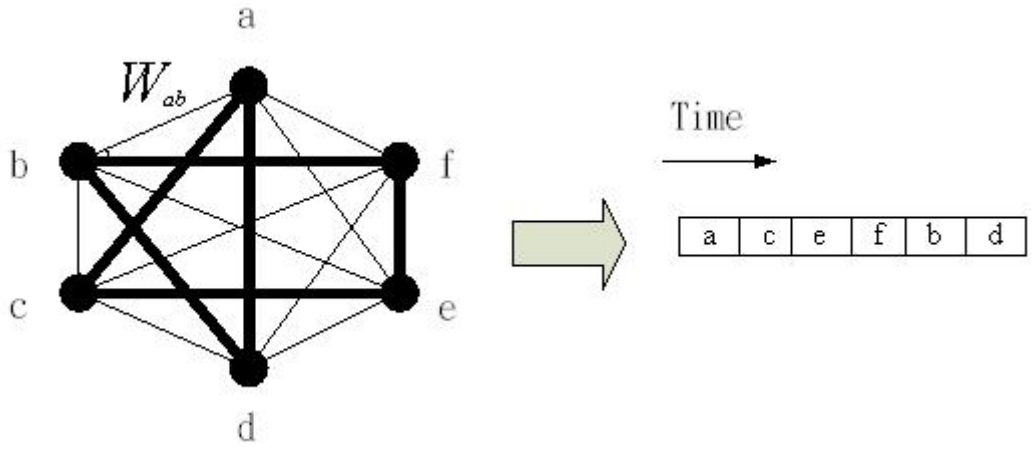


圖 6 CBS 置換策略

第三章　問題描述

在先前的章節，我們已經介紹了許多相關的演算法及其運作方式。再接下來的這一章節裡我們會敘述在先前的文獻探討裡所忽略的問題，本章節將分節介紹無線廣播的環境與運作模式，快取伺服器裡的資料項排程問題。

第一節　無線廣播的環境與運作模式

在無線廣播的環境下，資料項的傳送不用藉由實體網路來傳送，所以客戶端的機動性是可以非常大的，但是在無線廣播的環境裡一直有一個問題存在，那就是網路頻寬的問題。到目前為止相對於有線網路的傳輸速度最高可達1*Gbps* ，但無線網路的頻寬最快卻只能達到108 *Mbps*。因此如何在有限的頻寬下，將資料傳送給客戶端就成了眾多研究者討論的方向。

在無線廣播的環境中由於頻寬的限制，資料項的傳送大多採取廣播的方式來傳遞資料，因此客戶端必須隨時監聽廣播頻道以接收本身所需求的資料，以防止資料錯失得花上多餘的時間再重新等待一次。無線廣播環境中，客戶端的運作主要是藉由電池來提供電力，所以必須考量到電力的問題，而且當客戶端監聽廣播頻道時會消耗大量的電力。因此在無線網路中，除了考量網路頻寬之外還必須考慮電力問題。由以上的限

制條件，可以了解在無線廣播環境下所要探討的目的為「在有限的頻寬和電力供應下，如何能有效降低客戶端的平均等待時間」。

但是在無線廣播環境中，伺服器必須根據客戶端所提出的資料項請求來決定資料項的廣播順序，此一廣播順序稱為單一廣播週期。而不同的快取置換策略所得到的排列順序也會有所不同，但不論是使用何種快取置換策略來進行資料排程，其最終目的都是為了降低客戶端整體的平均等待時間。

第二節　快取伺服器裡的資料排程議題

因為無線網路的頻寬是有限的，遠不及有線網路的頻寬，因此快取伺服器扮演很重要的角色。在proxy cache的環境裡，當使用者查詢請求資料項時會先向鄰近的快取伺服器查詢，假使查詢的資料項有儲存在快取伺服器中，客戶端請求的資料項能快速的被滿足；相反的，若鄰近的伺服器沒有客戶端所請求的資料，客戶端的請求只能透過傳送的方式向遠端的資料庫伺服器端發出資料項的請求，再轉送給有需求的客戶端，在長久的等待時間之下，會使客戶端的負載產生問題。

第三節　快取伺服器環境

為了能讓所提出的方案更適用於真實環境中，我們應該努力將環境

模擬與現實環境相似。因此，在 server 端裡的快取伺服器內的資料項被讀取率應該與真實環境中相似。

本論文所謂的快取伺服器環境，指的是：在廣播系統裡，客戶端查詢資料項時，會先向鄰近的快取伺服器查詢是否有查詢的資料項，若沒有此資料項再向遠端資料庫伺服器發出查詢的請求，當資料庫伺服器廣播此資料項，快取伺服器將此資料項接收後，檢查此資料項是否已儲存在快取伺服器中，若已儲存在快取伺服器中則更新此資料項的被讀取時間，若未儲存在快取伺服器中，則選擇置換儲存在快取中的某一資料項。因此，當快取伺服器接收到廣播資料項，置換存於快取伺服器的資料項，以降低客戶端平均查詢讀取時間是本篇論文探討的主題。

(例1)

令 d_i 為資料項i，如圖7所示，假設Server正在廣播資料項 d_4，而目前在快取伺服的快取中存在 d_0、d_1、d_2、d_3，因為 d_4 沒有儲存在快取中，因此得需要置換快取中的資料項，期望能降低客戶端的平均讀取時間。

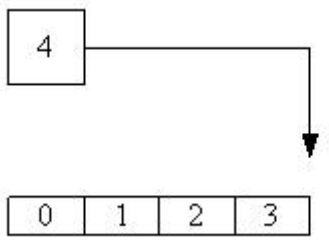


圖 7 快取伺服器接收到廣播資料項 d_4

在無線環境的快取伺服器裡，置換快取伺服器裡的資料項有許多的方法，本論文探討的主題是該置換如何置換儲存在快取伺服器的資料項，以期望降低平均查詢讀取時間。

在第二章節的文獻中提及的LRU、LSU、Size policy、CF、SF的置換方法只考慮單一資料項的被請求時間、請求次數、資料項大小，找到應該置換的資料項，其中並沒有考量到資料項之間的可能關聯性。CBS[8]，CBS利用資料項之間的權重值來決定快取伺服器的資料項儲存順序，其中CBS[8]的方法只考慮到二個資料項之間的關聯性，無法應用在多個資料項之間的關聯性。我們提出在此最小需求置換演算法(MD)，目的在於能減輕server快取伺服器的負載，增加工作效率，並且能有效降低使用者發出查詢的平均等待時間。

第四章　　最小需求置換演算法

最小需求置換演算法可以有效解決其他演算法 LRU[2]、CF[4]與 SF 忽略資料關聯性的問題，也可以解決 CBS[8] 未考慮多資料項之間的關聯性問題。本研究藉由提出此演算法以降低客戶端的平均查詢讀取時間。以下將完整詳細說明最小需求置換演算法的運作。

第一節　環境架構

在無線網路環境裡，假設客戶端附近會有一個本地快取伺服器，當客戶端請求資料項時，會先查詢本地快取伺服器是否有查詢的資料項，快取伺服器裡若有此資料項時，則可以快速降低客戶端的請求；若快取伺服器沒有客戶端查詢的資料項，則得向遠端資料庫 server 發出請求並由遠端資料庫 server 將資料項廣播給本地的快取伺服器，藉以提高本地快取伺服器的效能，降低客戶端的平均查詢讀取時間

第二節　最小需求置換演算法概念

如之前所敘述，最小需求置換演算法目的為有效解決客戶端的平均等待時間。最小需求置換演算法考慮資料項與資料項之間彼此的關聯性，比較此資料項的需求值，找出需求值最小的資料項，將接收到的廣播值置換於快取中需求值最小資料項的儲存位置。最小需求值的意義為

在快取中的資料項被查詢的頻率總和，最小需求值越高者，代表資料項被查詢的頻率越高。因此，我們選擇需求值最小的資料項置換成目前接收到的廣播值。

當快取伺服器接收到遠端資料庫 server 廣播資料項時，快取伺服器需重新更新每一資料項需求值，從這些資料項中找尋最小需求值最小的資料項，並置換成目前接收到的廣播值。

因為冷門的資料項較不常被廣播伺服器廣播，如果未接收此資料項，客戶端得在下一次廣播週期時才能接收到此資料項，造成查詢讀取時間可能會過於冗長。因此，如何有效率的置換快取伺服器資料項，以降低客戶端的查詢讀取時間為本研究之重點。為了降低客戶端的平均查詢讀取時間，我們考量資料項之間的關聯性。在快取伺服器中，除了考量 query 的頻率之外，同時也考量每個資料項的需求值。當快取伺服器接收到遠端資料庫 server 廣播資料項時，快取中資料項的需求值重新計算，從這些資料項中找尋需求值最小的資料項，將目前的廣播值資料項置換具有最小需求值的資料項，以達到有效率的降低客戶端的查詢讀取時間。

第三節　解決方案概述流程圖

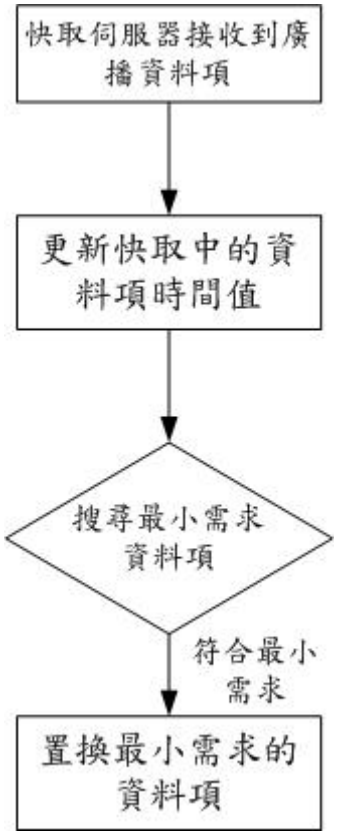


圖 8 最小需求置換演算法流程圖

假設廣播伺服器正在廣播資料項d_k，快取伺服器會先逐一檢查在快取伺服器中的資料項，從第一個資料項開始檢查，直到檢查完最後一個資料項為止，若接收到的d_k已經儲存在快取伺服器中，則更新此該資料項被讀取的時間；若d_k沒有儲存在快取伺服器裡面，則由資料項d_k開始往前尋找未存於快取的資料。

第四節　運作程序

在文獻探討中所提出之方法皆是以往後搜尋為主，其中並沒有探討到資料的關聯性，直到CBS[7]，CBS的方法是從接收到廣播值之後開始尋找資料的關聯性，本文所提出的方法不同於以往後搜尋的作法，而是採取往前搜尋資料項與資料項之間的關聯性。

4.4.1 運算定義

為了能更清楚的說明問題與本研究內容的呈現，表格 4 定義本研究所使用的符號。

表格 2 本研究所提用之演算法將使用以下定義

參數	參數定義
d_i	資料項 i
q_i	查詢請求 i
C	目前儲存在快取伺服器的資料項集合。
D(C)	當快取伺服器的集合為 C 時，其平均查詢讀取時間
τ_i	在目前的廣播序列下，查詢q_i的最後一個不在快取中的資料項廣播時間與現在的時間差。
f_i	q_i的查詢請求頻率

4.4.1.1 平均查詢讀取時間

下列為演算法中使用到的公式說明:

D(C)為q_i的平均查詢讀取時間:

$$D(C)= \frac{\sum f_i \tau_j}{\sum_i f_i} \qquad \text{公式(3)}$$

(例 3)

假設遠端資料庫伺服器不斷重複廣播d_1到d_6的資料項，儲存於快取伺服器的資料項 C=$\{d_1、d_3、d_5\}$ 令q_1=($d_2、d_3、d_5$)，且其需求頻率為f_1=30，目前廣播資料項d_6，由於q_1查詢的資料項中d_2不存於快取中，因此得等廣播序列再次廣播d_2時才能接收d_2，q_1的查詢才能滿足。因此q_1的等待時間τ_1=2，其查詢讀取時間為:

$$D(C)= \frac{\sum f_j \tau_j}{\sum_j f_i} = D(\{d_1、d_3、d_5\})=\frac{30\times 2}{30} = 2$$

4.4.1.2 最小快取需求(MinDemand):

位於伺服器的快取中之資料項，會因不斷更新而變動，為了確保資料項皆為客戶端所需，藉由置換機制更新伺服器的快取，使得客戶端的查詢讀取時間得以降低，為本研究所要探討。本研究藉由尋找在快取伺服器中最小需求的資料項，將目前的廣播值資料項置換具有最小需求值的資料項。

最小需求置換演算法符號定義如下:

- *π(j，k)*代表從 server 端重複不斷的廣播中之序列，從資料項d_j到d_k所有資料項集合。

$$\pi(j,k)=\begin{cases}\{d_j,d_{j+1},d_{j+2},\ldots,d_k\} & \text{if}(k\geq j)\\ \{d_k,d_{k+1},d_{k+2},\ldots,d_n,d_1,d_2,\ldots,d_j\} & \text{if}(j>k)\end{cases}$$

- *Include（i，j）*:判斷q_i是否有包含資料項d_j，

$$\text{Include}(i,j)=\begin{cases}1 & :\text{如果}q_i\text{包含資料項}d_j\\ 0 & :\text{Otherwise}\end{cases}$$

- NotInCache(*i，S*):判斷q_i所包含的任一資料項是否存在於集合 S 中

$$\text{NotInCache}(i,S)=\begin{cases}1 & :q_i\text{中至少包含一個不在快取伺服中的資料項}\\ & \quad\text{且此資料項在S的集合中}\\ 0 & :\text{Otherwise}\end{cases}$$

當快取內的集合為 C，且目前廣播的廣播序列中，正在廣播資料項d_k，則在快取中的資料項d_j其需求值為:

$$\text{Demand}_j(k,C)=\sum_i f_i\times\text{Include}(i,j)-\sum_i f_i\times\text{Include}(i,j)\times\text{NotInCache}(i,\pi(j,k)) \quad \text{(公式 4)}$$

其中$\sum_i f_i$*Include(i，j)為包含資料項d_j的所有查詢頻率總和，若包含資料項d_j的查詢已至少存在一個不在快取的資料項，且此資料項的下次

廣播時間在資料項d_j的廣播時間之後，則將此查詢的請求頻率從第一項的總和中減去。則此頻率總和為$\sum_i f_i$*Include(i，j)*NotInCache(i，π(j，k))

找尋最小需求的主要的原因在於q_i在請求資料項時，除了請求儲存在快取伺服器的資料項，還需請求其他未儲存在快取伺服器的資料項，所以我們的方法選擇先置換q_i請求的資料項中，有儲存在快取伺服器的資料項，等待廣播伺服器下一次廣播此資料項時再接收。

4.4.2 最小需求置換演算步驟

1. 更新快取中的資料項時間值:

假設有一個無線廣播的資料項排程為$d_1, d_2, d_3, \ldots, \ldots, d_m$且一直不斷依序循環廣播，當遠端資料庫伺服器正在廣播資料項d_k時，檢查d_k是否已存在於快取中，若已存在則更新此d_k的被讀取時間；若沒有存在於快取中，則進行下一步驟。

2.搜尋在快取中的最小需求資料項d_i:

令 *Delay* [i]為q_i的查詢請求是否延遲，1≦i≦m，選擇某一個 i 的符合下列的條件。

- *Delay*[i]= $\begin{cases} 1 & \text{:快取中至少已不存在一個在q(i)中的資料項} \\ 0 & \text{:0therwise} \end{cases}$

首先清空 Delay 的矩陣，將 *Delay*[i]的值清空為 0，再往前檢查d_{k-1}是

否存於快取伺服器中，若d_{k-1}存在快取伺服器中，則再往前檢查d_{k-2}，直到找出不存於快取伺服器的資料項。由廣播序列中可以看出，在d_k之前一個不在快取伺服器的資料項為d_ℓ，尋找所有查詢d_ℓ的 query，針對這些 query 查詢的資料項中，找出存於快取伺服器內廣播序列小於d_ℓ之前的資料項，並從此些資料項需求值中，進而比較出需求值最小的資料項d_i。

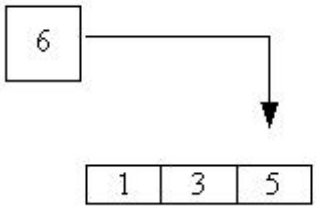
3.將資料項d_i置換成接收到的廣播值d_k:

將此d_i置換成目前接收到的廣播值d_k。因為在此些查詢中至少已有一個資料項d_ℓ不在快取中，置換此資料項不會影響這個 query 的查詢讀取時間。

為了說明本演算法的運作方式，以下列一個案例推導

(例 4)

假設遠端資料庫伺服器不斷重複廣播d_1到d_6的資料項，快取中目前儲存的資料項為 C=$\{d_1$、d_3、$d_5\}$且q_1=(d_1、d_2、d_5)、q_2=(d_1、d_2、d_3)、q_3=(d_1、d_4、d_5)、q_4=(d_2、d_3、d_4)、q_5=(d_5、d_6)，對應的需求頻率為f_1=30、f_2=22、f_3=19、f_4=14、f_5=2。其等待時間為τ_1=2、τ_2=2、τ_3=4、τ_4=4、τ_5=6。圖 10 表示假設快取伺服器接收到從遠端資料庫 server 廣播的資料項d_6



圖表 9 快取伺服器接收到廣播資料項d_6

D(d_1、d_3、d_5)=$\frac{2\times30}{87}+\frac{2\times22}{87}+\frac{4\times19}{87}+\frac{4\times14}{87}+\frac{6\times2}{87}=2.85$，由公式 3 可以得知客戶端的平均查詢讀取時間為 2.85。

4.4.2.1 MD 置換策略

由圖10可以得知，快取伺服器接收到從遠端資料庫server廣播的資料項d_6，檢查d_6是否有存在快取伺服器裡，若有儲存在快取伺服器裡則更新d_6的被讀取時間，若沒有儲存在快取伺服器則找尋最小需求值來置換成目前廣播資料項。

步驟一，清空 Delay 矩陣

在本案例中先令 *Delay*[1]=0、*Delay*[2]=0、*Delay*[3]=0、*Delay*[4]=0、*Delay*[5]=0。再者把有查詢請求資料項d_1、d_3、d_5的所有頻率加總

$Demand_1=f_1+f_2+f_3$=30+22+19=71

$Demand_3=f_2+f_4$=22+14=36

$Demand_5=f_1+f_3+f_5$=30+19+2=51

步驟二，由廣播序列中可以看出，在d_6之前一個不在快取伺服器的資料項為d_4，查詢請求資料項d_4的第一個 query 為q_3，q_3=(d_1、d_4、d_5)，確認q_3的請求是否有延遲， *Delay*[3]=0 代表的請求未延遲。由公式 4 可以得知*Demand* $_1$ 的需求更改為:

Demand $_1$ =71-19×1=52

q_3=(d_1、d_4、d_5)，因為d_4不在快取中，所以在廣播序列之前的資料項d_1，其是否存在快取中並不會影響q_3的查詢讀取時間，因此在d_1的查詢頻率總和中，將q_3的頻率移除。此時設定 *Delay*[3]為 1 表示此時之後在快取伺服器中已不存在一個q_3所包含的資料項d_4，所以q_3的查詢不再被考慮其頻率。

查詢請求資料項d_4的第二個 query 為q_4，q_4=(d_2、d_3、d_4)，確認q_4的請求是否有延遲， *Delay*[4]=0 代表q_4的請求未延遲。由公式 4 可以得知*Demand* $_3$ 的需求為:

Demand $_3$ =36-14× 1=22

q_4=(d_2、d_3、d_4)，因為d_4不在快取中，所以在廣播序列之前的資料項d_3，其是否存在快取中並不會影響q_4的查詢讀取時間，因此在d_3的查詢頻率總和中，將q_4的頻率移除。此時設定 *Delay*[4]=1 表示此時在快取伺服器中已不存在一個q_4所包含的資料項d_4，所以q_4的查詢不再被考

慮其頻率。

由廣播序列中可以看出，在d_4之前一個不在快取中的資料項為d_2，查詢請求資料項d_2的第一個 query 為q_1，q_1=(d_1、d_2、d_5)，確認q_1的請求是否有延遲， *Delay*(1)=0 代表q_1的請求未延遲。由公式 4 可以得知 $Demand_1$ 的需求為:

$Demand_1$=52-30×1=22

q_1=(d_1、d_2、d_5)，因為d_2不在快取中，所以在廣播序列之前的資料項d_1，其是否存在快取中並不會影響q_4的查詢讀取時間，因此在d_1的查詢頻率總和中，將q_1的頻率移除。此時設定 *Dela*y(1)=1 表示此時在快取伺服器中已不存在一個q_1所包含的資料項d_2，所以q_1的查詢不再被考慮其頻率。

查詢請求資料項d_2的第一個 query 為q_2，q_2=(d_1、d_2、d_3)，確認q_2的請求是否有延遲， *Delay*(2)=0 代表q_2的請求未延遲。由公式 4 可以得知 $Demand_1$ 的需求為:

$Demand_1$=22-22×1=0

q_2=(d_1、d_2、d_3)，因為d_2不在快取中，所以在廣播序列之前的資料項d_1，其是否存在快取中並不會影響q_2的查詢讀取時間，因此在d_1的查詢頻率總和中，將q_2的頻率移除。此時設定 *Dela*y(2)=1 表示此時在快

取伺服器中已不存在一個q_2所包含的資料項d_2，所以q_2的查詢不再被考慮其頻率。

步驟三: $Demand_1$=0、$Demand_3$=22、$Demand_5$=51，比較之後可以得知d_1的需求值最小，因此我們選擇將廣播資料項d_6置換於快取中d_1的位置。

圖 10 最小需求置換策略

$$D((C/\{d_1\})\cup\{d_6\})=D(d_3、d_5、d_6)=\frac{2\times30}{87}+\frac{2\times22}{87}+\frac{4\times19}{87}+\frac{4\times14}{87}+\frac{0\times2}{87}=2.71$$

第五節　結論

藉由本實驗的案例推導，可以清楚的了解 MD 的實驗流程， MD 的方法藉由尋找資料項與資料項之間的關聯性，因此在某些情形下能夠比 LRU 更有效率的找到該置換的資料項，有效地降低客戶端的平均查詢讀取時間，在找尋資料項時也不會浪費過多的時間。

第五章　效能評估

本研究主要目的是探討在快取伺服器內，考慮廣播資料項的循環週期，並考量廣播資料項間彼此之間關聯性，提供一個較佳的無線廣播資料排程，有效率的降低客戶端的平均等待時間。爲了證實本文提出的往前搜尋法能有效改善客戶端的平均等待時間，必須藉由客觀的實驗分析證明。本章節分為四小節，第一節為模擬環境介紹，第二節為實驗環境的資料項，第三節為實驗環境分析，第四節為結論。

第一節　模擬環境介紹

在本次的模擬實驗中，在演算法的部份使用JAVA程式語言做為我們的演算法程式語言。在模擬環境的中，本研究假設客戶端請求的資料項大小相等。在快取伺服器方面，快取伺服器在每次接收到新的廣播資料項時做檢查的動作，若新的資料項沒有儲存在快取伺服器裡面，再將資料項依據往前搜尋法來進行快取置換策略；在進行置換策略時，每接收一個新的廣播資料項，就會在快取伺服器裡面做檢查的動作。因此每一資料項在快取伺服器裡面將只檢查一次，資料項的置換只透過一個廣播頻道。

第二節　實驗環境的資料項

在實驗環境模擬過程中，在客戶端所產生的資料項請求部份，請求(request)資料項為隨機分配，沒有次序的。我們採用 Zipf 分配 [9]來假設客戶端讀取資料項的機率。θ 為偏斜因素，θ 值介於 0 到∞ 之間，當 θ 從 0 增加到∞資料存取的頻率偏斜程度會越來越高。當 θ=0 代表客戶端存取資料項的頻率均勻分配，讀取每一個資料項的頻率幾盡相同。當 θ=∞資料項存取會呈現極為偏差的 Zipf 分佈而且資料項的請求頻率也會不平均。

實驗的資料主要是模擬客戶端對快取伺服器可能提出資料項的請求。因此在本研究的實驗環境資料項中，主要考慮因素有廣播的資料項個數，廣播query的個數以及廣播資料的偏斜度大小等因素；

表 3　實驗環境資料檔

參數名稱	代表的意思
Cachesize	20，30，40
Number of Data Item	100
query	200
sel	2
Zipf θ	0.2、0.4、0.6、0.8

我們選擇的實驗環境為快取伺服器大小為20、30、40個資料項，100的資料項，query數200個，每個query2個資料項，Zipf θ為0.2、0.4、0.6、0.8。

第三節　實驗環境分析

在第一個實驗中，主要討論MD與LRU的平均等待時間，假設廣播資料項為100 個，query個數為200 個，快取伺服器的大小為20個資料項，計算平均等待時間的偏斜度分別為0.2、0.4、0.6、0.8。

表 4 快取伺服器大小為20的實驗數據

偏斜度	LRU	MD
0.2	73.11114136	62.62857721
0.4	71.94764179	62.09015196
0.6	69.78142824	61.53948843
0.8	63.31454407	60.84342707

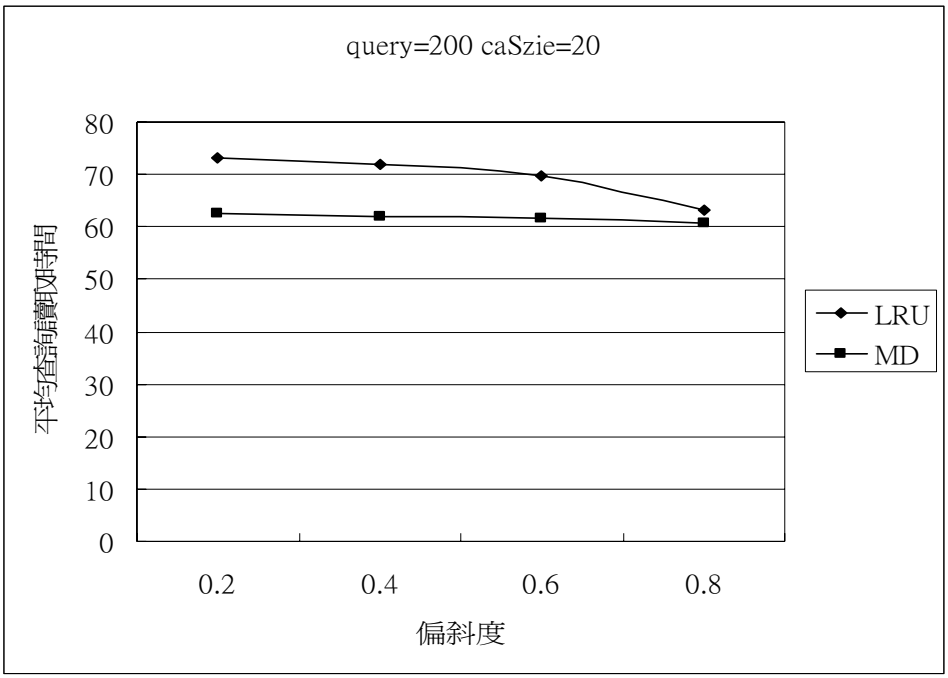


圖 11 當快取伺服器的大小為 20

圖 11 表示，當 query=200、Cachsize=20、偏斜度為 0.2 的時候，我們的方法 MD 能夠有效的降低使用者的等待時間，與 LRU 的差距為 10.4829，當偏斜度為 0.8 時，MD 也可以比 LRU 能有效降低客戶端的平均查詢讀取時間。

在第二個實驗中，也是在討論MD與LRU的平均等待時間，假設廣播資料項為100 個，query個數為200 個，快取伺服器的大小為30個資料項，計算平均等待時間的偏斜度分別為0.2、0.4、0.6、0.8。

表 5 快取伺服器大小為30的實驗數據

	LRU	MD
0.2	68.54217113	55.24743315
0.4	66.94797945	55.13993402
0.6	63.26571444	53.17472337
0.8	57.43803609	52.91638332

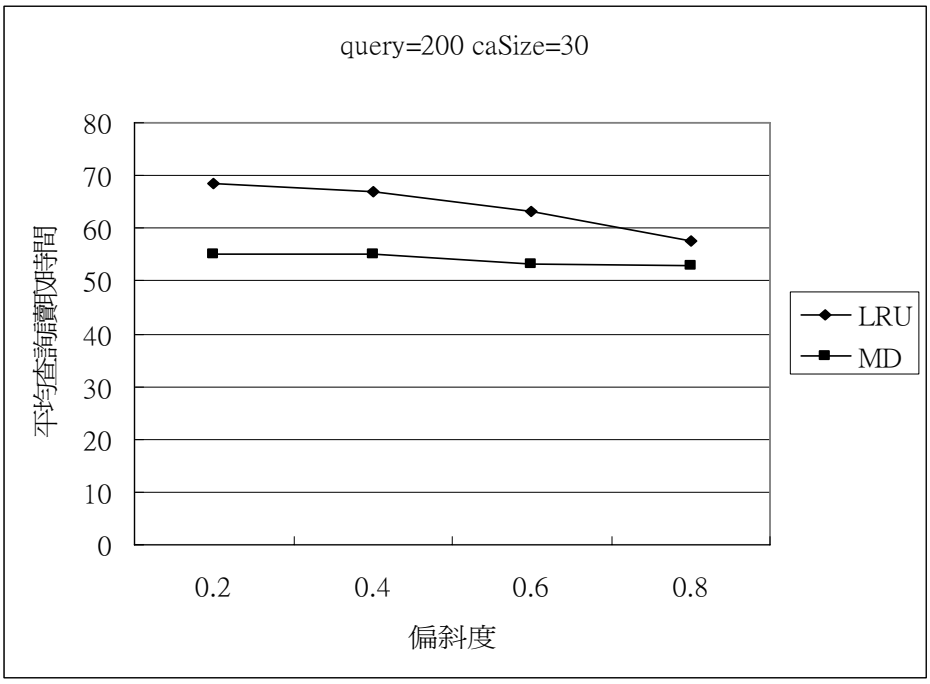


圖 12 當快取伺服器大小為 30

圖 12 表示，當 query=200、Cachsize=30、偏斜度=0.2 的時候，我們的方法 MD 能有效的降低使用者的等待時間，與 LRU 的差距為 13.2947，當偏斜度為 0.8 時，MD 也可以比 LRU 能有效降低客戶端的平均查詢讀取時間。

在第三個實驗中，也是在討論MD與LRU的平均等待時間，假設廣播

資料項為100 個，query個數為200 個，快取伺服器的大小為30個資料項，計算平均等待時間的偏斜度分別為0.2、0.4、0.6、0.8。

表 6 當快取伺服器為40的實驗數據

	LRU	MD
0.2	62.48064388	48.0668654
0.4	61.194423	49.9888243
0.6	57.64516164	48.6829155
0.8	51.87369518	47.8863993

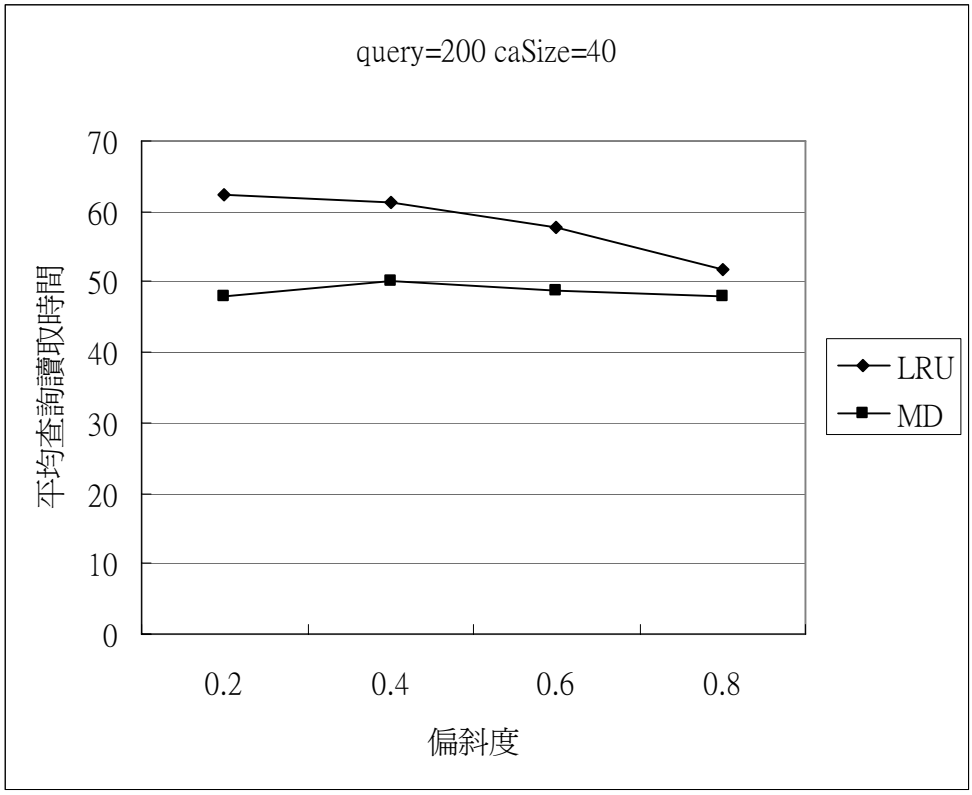


圖 13 當快取伺服器大小為 40

圖 13 表示，在此時當 query=200、Cachsize=40、偏斜度=0.2 的時候，我們的方法 MD 能夠更有效的降低使用者的等待時間。與 LRU 的差距為

14.4138，當偏斜度為 0.8 時，MD 也可以比 LRU 能有效降低客戶端的平均查詢讀取時間。由實驗結果得以得知當 CacheSize 較大時，MD 可以比 LRU 更有效的降低平均查詢讀取時間。

第四節　實驗結果探討

我們所提出的最小需求置換演算法主要的目的是在資料項平均分布的時候改善 LRU 的平均等待時間，由實驗結果可以得知在快取伺服器的資料項只能儲存 20 個資料項時，LRU 的平均等待時間是 74.111 而我們的方法可以降低平均等待時間為 62.6285，在偏斜度為 0.8 的時候 LRU 的平均等待時間為 63.3145，我們的 MD 演算法為 60.8434。在快取伺服器的資料項增加 30 個資料項時，LRU 的平均等待時間是 64.5821 而我們的方法可以降低平均等待時間為 55.2474，在偏斜度為 0.8 的時候 LRU 的平均等待時間為 57.4380，我們的 MD 演算法為 52.9164。但是當快取伺服器的資料項增加到 40 個資料項時，我們的方法更能有效的降低平均等待時間，LRU 的平均等待時間為 62.4806 我們的方法為 48.0668，在偏斜度為 0.8 的時候 LRU 的平均等待時間為 51.8737，我們的 MD 演算法為 47.8864。

由我們的實驗可以得知我們所提出的最小需求置換演算法(MD)可以有效的降低客戶端在請求資料時的平均等待時間。

第六章　結論

以往的研究著重在「單一資料」的廣播快取伺服器環境，本篇論文，主要討論在「資料關聯性」之要求模式的廣播快取伺服器環境，然而根據本實驗的分析，論文中對於以往使用的要求模式的廣播演算法討論了其不適應在「資料關聯性」的環境，並由實驗顯示以往的要求模式的排程演算法不適用在「資料關聯性」廣播環境，需要考慮新的判斷因素。

以往的 LRU 只是單純的做搜尋的動作，沒有考慮到資料項與資料項之間的關連性，因此我們提出一個新的要求模式的排程演算法「MD」，其考慮了考慮到資料項與資料項之間的關連性，能夠滿足客戶端請求的數目，這兩個重要的因素關係著「資料關聯性」廣播的效能，可以有效率地降低客戶端的平均等待時間。在資料種類大量的環境下，我們提出了減少計算的機制來減少比對資訊種類的數目，又能降低了客戶端的等待時間。

參考文獻

中文部分:

[1] 陳佳慧，「WWW代理伺服器的部份快取置換策略」，元智大學資訊工程研究所碩士論文，民國89年。

西文部份:

[2] M. Abrams, R. Standridge, G. Abdulla, S. Williams, and A Edward, “Fox. Caching Proxies: Limitations and Potentials,” In Proceedings of the 4th International WWW Conference, July 1995.

[3] M. Abrams, G. Abdulla, S. Williams, and A. Edward Fox, “Removal Policies in NetWork Caches for WWW Documents,” In Proceedings of the ACM SIGCOMM96, August 1996.

[4] S. Acharya, M. Franklin, and S. Zdonik, “Prefetch from a broadcast disk.” In Proceedings of the International Conference on Data Engineering,1996.

[5] Y.I. Chang, W.H. Hsieh, “An Efficient Scheduling Method for Query-Set-based Broadcasting in Mobile Environments,” In IEEE Conf. Proceedings of the 24th International Conference on Distributed Computing Systems Workshops, 2004.

[6] A. Demet and F. Michael, “R*W: A Scheduling Approach for Large-Scale On-Demand Data Broadcast” IEEE/ACM Transaction on Networking, VOL. 7, NO. 6, DECEMBER 1999.

[7] John Dilley, Martin Arlitt, and Stephane Perret, “Nhancement and Validation of Squid’s Cache Replacement Policy,” In Proceedings of the 4th International WWW Caching Workshop, April 1999.

[8] Y. Etsuko, H. Takahiro, T. Masahiko and N. Shojiro, “Scheduling and

Caching Strategies for Broadcasting Correlated Data," ACM, 2001.

[9] L. C. Kuo, and W. H. Shu, "An Ad Hoc On-Demand Routing Protocol with High Packet Delivery Fraction," In IEEE Conf. International Conference on Mobile Ad-hoc and Sensor Systems, pp. 594-596, 2004.

[10] S. Navrati, B.Kalyan and K. D. Sajal,"Design and Performance Analysis of a Dynamic Hybrid Scheduling Algorithm for Heterogeneous Asymmetric Environments," In IEEE Conf. Proceedings of the 18th International Parallel and Distributed Processing Symposium, 2004.

[11] W.W. Sun, W.B. Shi, B. Shi , "A cost-efficient scheduling algorithm of on-demand broadcasts," Manufactured in The Netherlands, Wireless Networks 9, 239–247, 2003.

[12] Z. Wang and C. Jon, "Prefetching in World Wide Web," In IEEE Globecom'96, 1996.

[13] H.W. Wei, P.C. Huang , H.P. Chang , W.K. Shih, "Scheduling Real-Time Information in a Broadcast System with Non-Real-Time Information," In IEEE Conf. Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, 2005.

[14] T. J. Wirawan, F. C. Heng, L. B. Sung, "An Efficient Scheduling Scheme for High Speed IEEE 802.11 WLANs," In IEEE Conf. pp. 2589-2593, 2003.

[15] R.P. Wooster, "Optimizing Response Time, Rather than Hit Rates, of WWW Proxy Cache," Master's thesis, Virginia Polytechnic Institute and State University,Blacksburg, Virginia, December, 1996.

[16] J. Z. Wang ,Z. Du, P. K. Srimani, "Network Cache Model for Wireless Proxy Caching" In Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2005.

[17] Y.Wu and G. Cao, "Stretch-Optimal Scheduling for On-Demand Data Broadcasts," In IEEE Proceeding of the IEEE International Conference on Computer Communications and Networks, pp.500-504, 2001.

[18]W. Xiao and C. S. Victor, “Preemptive Maximum Stretch Optimization Scheduling for Wireless On-Demand Data Broadcast,” In IEEE Proc. Proceedings of the International Database Engineering and Applications Symposium, 2004.