南 華 大 學

資訊管理學系

碩士論文

在要求式的廣播環境中
以網路編碼為基礎的有時效性資料廣播
Data Broadcasting with Deadline Based on Network Coding in On-demand Broadcast Environment

研 究 生：黃 雲 賢

指 導 教 授：吳 光 閔 博士

中華民國　　九十八年　　十二月

南　華　大　學

資訊管理學系(所)

碩 士 學 位 論 文

在要求式的廣播環境中

以網路編碼為基礎的有時效性資料廣播

研究生：黃雲賢

經考試合格特此證明

口試委員：楊庚european

吳光陽

蔡德諱

指導教授：吳光陽

系主任(所長)：鍾國賢

口試日期：中華民國　九十八年　十二月　十八日

南華大學資訊管理學系碩士論文著作財產權同意書

立書人：黃雲賢 之碩士畢業論文

中文題目：在要求式的廣播環境中，以網路編碼為基礎的有時效性資料廣播

英文題目：Data Broadcasting with Deadline Based on Network Coding in On-demand Broadcast Environment

指導教授：吳光閔 博士

學生與指導老師就本篇論文內容及資料其著作財產權歸屬如下：

☑ 共同享有著作權

☐ 共同享有著作權，學生願「拋棄」著作財產權

☐ 學生獨自享有著作財產權

學　　生：黃雲賢 （請親自簽名）

指導老師：吳光閔 （請親自簽名）

中　華　民　國　九十八　年　　十二　月　十八　日

南 華 大 學 碩 士 班 研 究 生

論 文 指 導 教 授 推 薦 函

資訊管理 系碩士班 黃雲賢 君所提之論文

在要求式的廣播環境中，以網路編碼為基礎的有時效性資料廣播

係由本人指導撰述，同意提付審查。

指導教授 吳光閔

99 年 1 月 4 日

誌　　謝

在研究所的這段時間過的非常快，還記得剛入學時懵懵懂懂的我，如今已完成論文也即將完成碩士學位的攻讀。非常感謝慶豐、詠生、淳淳學長姐對我的教導，使我在剛踏入這塊研究領域時不會徬徨無助，可以很快的調適過來。非常感謝 佳濬、芸蓁、弘文、鈺凱、躍進、士峰以及全班同學的指導與照顧，當我在課業與生活上遇到困難時給我即時的幫助，還要感謝東迪學弟對我的鼓勵，以及女朋友宜芳對我的支持。謝謝大家的幫助與鼓勵，讓我可以順利完成研究所的課業。

最後要特別感謝我的指導教授吳光閔老師與蔡德謙老師的細心教導，透過老師您的教導讓我在遇到論文的瓶頸時總能給我適時的幫助。在與老師相處的過程當中，我學習到老師您嚴謹務實的研究精神與耐心關愛每一位學生的生活態度，讓學生獲益良多。希望將來能有所成就不辜負老師的栽培，對老師有太多的感謝，在此學生雲賢要誠心的跟老師您說聲：老師謝謝您，您辛苦了。

在要求式的廣播環境中，以網路編碼為基礎的有時效性資料廣播

學生：黃雲賢　　指導教授：吳光閔

南　華　大　學　資訊管理學系

摘　　要

在資料廣播環境的架構下，要求式廣播是一個有效率的無線資料傳輸方式。在真實的廣播環境中，使用者所發出的請求通常是包含多個資料項的，因此近來有很多相關的研究開始在探討多個資料項請求的廣播排程。然而，傳統的要求式資料廣播排程假設每一個廣播時間單位（one time slot）只能存放一個資料項，因此一旦使用者錯過所需要的資料項，則必須要等待下一個廣播週期才能收到所需要的資料項。在這裡本研究採用了一個網路編碼的技術，將多個資料項編碼成一個編碼資料項，配置到一個廣播時間單位上並且資料項可以重複出現在不同的排程位置，因此一個廣播時間單位就有可以服務不同資料項的請求，要是使用者錯過所需要的資料項時，就可以在不同排程位置上，利用本身已經 Cache 的資料項來解碼，就可以獲得所需要的資料項。在本研究中，我們以網路編碼為基礎並且考量了請求的時效性，提出了一個新的衡量標準來決定哪一個請求要優先被服務，其計算出來的權重值越高的越優先服務。根據我們的方法與[3]比較的實驗結果顯示，我們的方法在平均存取時間上只多了 5%的存取時間，但是在請求成功的比率上，我們的方法提高了 20％的請求成功比率。

關鍵字： 要求式廣播，廣播排程，多資料項的請求，網路編碼

Data Broadcasting with Deadline Based on Network Coding in On-demand Broadcast Environment

Student：Yun-Shien Huang　　Advisors：Dr. Guang-Ming Wu.

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

ABSTRACT

On-demand broadcast is an effective wireless data dissemination technique in data broadcast environment. In real world, users' queries usually include multiple data items. Recently, there are many related papers began to explore multi-item queries of data scheduling. However, traditional on-demand data broadcast scheduling assume that each time slot includes only one data item. Therefore, the above constraint demands a mobile user to wait until the next broadcast cycle to retrieve the data item if he misses the item in the cycle. Using network coding technique, server can encode and broadcast multiple data items in a time slot. The data item was repeated allocated different positions in the broadcast cycle. A user can retrieve data items which they needed on the places in a broadcast cycle by using the user's cached data items to decode it. In this paper, we based on network coding model and we propose a new weight metrics that decided which query should be service with the highest priority. The metrics considers the factor of requests deadline to reduce deadline miss rate. Experiment results show that our algorithm can increase 20% success rate (which only increase 5% user's access time) compared with [3].

Keyword: On-demand Broadcast, Scheduling Broadcast, Multi-data Request, Network Coding

目　錄

表　　目　　錄

圖　　目　　錄

第一章、緒論

近年來隨著資訊科技的快速發展，使得人們享受到科技進步所帶來的便利與舒適。從 1990 年代至今，由於資訊科技產業的蓬勃發展與製造成本的降低加速了電腦的普及化，連帶的也帶動了網際網路的崛起與發展。網際網路隨著寬頻網路的時代來臨與電腦的普及化，充分地融入到每個人的日常生活當中，舉凡食衣住行皆可以在網路上搜尋到所需要的資訊甚至可以直接購買所需要的商品或服務，此外也經常透過網路來聯絡感情、傳遞與搜尋資料、觀看影片等等，網路不再只是單純的學術或軍事用途也落實了網路無國界、天涯若比鄰這些想法。

早期的網路傳輸速度慢、頻寬窄，因此資料的傳輸往往需要耗費相當多的等待時間，此外在使用上有相當多的限制，導致使用上相當的不便利。然而如今科技的進步加上高頻寬的時代來臨，大大的增加網際網路的易用性，加上現今無線網路（Wireless Network）的快速發展更是提高了網際網路的可攜性與使用率，也因此網路服務不再只有侷限在有線網路，人們可以透過無線網路進行無線上網，隨時隨地獲得所需要的資訊、商品或服務等等。

第一節　研究背景與動機

在早期的農業時代，人與人之間的訊息傳遞，必須透過口頭或者書信的方式來進行溝通，但是常因為人為或者表達不良等因素，而造成了誤解等紛爭。工業革命之後，交通工具與通訊設備的發明不僅縮短了人與人之間的距離也讓訊息的傳遞更迅速、正確。自 1990 年代資訊革命以來，資訊科技的進步早已是屢見不鮮，尤其是在網際網路興起之後，更是打破了時間的限制，人們可以利用電子郵件、即時通訊軟體等等來傳

遞即時的訊息與資訊，充分達到天捱若比鄰的想法；然而有線網路有著空間上的限制，換句話說也就是不能行動上網加上充滿競爭的社會環境與時間就是金錢的價值觀下，因此如何充分利用每一分每一秒並且隨時隨地獲得所需要的資訊、商品或服務的需求孕育而生，而無線網路的誕生正好滿足了上述的需求。無線網路可以隨時隨地的上網不受限於時間和空間的限制，譬如：在火車、汽車上可以透過手機進行行動上網查詢股市資訊，獲得所需要的股票資訊；因此無線網路的出現使得資訊的取得更方便並且與人們的生活更緊密的結合在一起。

隨著資訊科技的發展和進步，無線通訊（Wireless Communication）的應用已經相當的普遍，早已成為人類的日常生活當中不可或缺的一部份，如：PDA 手機所提供的衛星定位與行車路線地圖導覽服務。在無線網路的廣播環境裡，資料的傳輸通常是採用廣播的方式來滿足客戶端（Client）所發出的請求（Request），並且使 Client 端能在最短的時間裡獲得所需要的資訊。由於無線網路的移動性和方便性，使得無線網路的應用相當的廣泛，舉凡 PDA（Personal Digital Assist）、行動電話（Mobile Phone）、筆記型電腦（Notebook）等數位科技產品都可以找到應用無線網路的功能或服務。由上述得知無線廣播已經被廣泛的應用於資料的傳輸，從資訊系統到行動上網服務等等。無線廣播以更具體的方式來說，就是伺服器透過單一或多個頻道將使用者所需要的資料項準確遞送達到使用者的所在地，也就是所謂的 Client 端。一個廣播環境如圖 1-1 所示，其中主要的成員有 Client 端、無線網路、伺服器(Server)、資料庫（Database）。在 Client 端的使用者可以透過 PDA、行動電話、筆記型電腦等行動上網服務向伺服器發出請求來獲得所需要的資訊，伺服器接收到來自 Client 端的請求時，會依據 Client 端所需要的資料向資料庫擷取

資料並進行資料的排程，最後將排序好的資料排程透過無線網路來進行廣播以滿足客戶端的請求。

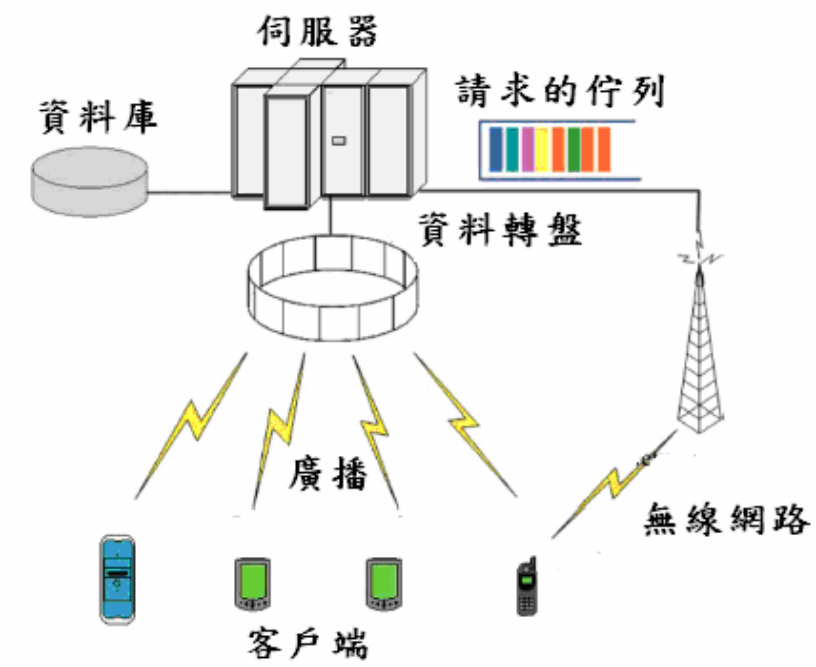

圖 1-1：廣播環境架構圖

在無線網路的發展上，無線廣播的技術仍然受限於頻寬、能源等因素，因此如何在有限的頻寬、降低能源的消耗之下將資料完整的傳送到客戶端並滿足最多的使用者將是很值得探討研究的。有很多的專家學者們分別討論不同的技術並將其應用在無線網路領域上，使得無線網路能在有限頻寬與能源下盡可能地去滿足絕大部分的 Client 端請求；此外在真實的環境中，資料項（Data Item）會隨著時間的不同而有不同的被點播率並且 Client 端所發出的請求也會具有不同的時效性（Deadline）。因此我們將要探討在一個要求式的廣播環境下（On-demand System），針對一個請求包含多個資料項與請求的資料項有冷熱門之分的先決條件之下，如何在請求的時效性之內將資料項進行一個有效率的廣播排程，使其能有效率地提高 Client 端請求被滿足的成功率(Success Rate)與降低 Client 端的平均存取時間(average successful request)。

第二節　研究主題

在無線網路的環境之下可以區分成許多不同類別的系統架構，分別適用在不同的無線網路環境裡，而不同的無線廣播環境其運作與特性也不盡相同。本研究專注在伺服器端及 Client 端的廣播排程，提出一個有效的廣播排程演算法，以服務較多使用者並降低 Client 端的等待時間並提高請求被滿足的成功率。在過去廣播排程的文獻中[2] [5] [6] [7] [11] [13] [15] [16]大都是以一個請求只能包含一個資料項作為其限制條件，考量單一個資料項來進行有效率的廣播排程。在真實的廣播環境中，一個請求可能是包含多個資料項並非只是單一個資料項而已，因此以包含多個資料項的請求為單位的廣播排程的相關研究[1] [4] [9] [12]孕育而生。上述的兩種廣播排程方面的研究，其主要是訴求降低 Client 端的平均存取時間和提高滿足請求的成功率，本文之研究亦是專注在這兩個主要訴求。

在上述的一般廣播排程中，分別提出不同的方法來安排廣播的資料項使其能在有限的頻寬下滿足最多的使用者，然而能滿足的請求數量相當有限，因此在文獻[3] [10]提出了一種結合 XOR 編碼技術的廣播排程方法，使其能在既有的有限頻寬之下，相較於之前的一般廣播排程方法滿足更多的使用者數量，但是文獻[3]中並沒有考量到 Client 端發出的請求通常是具有時效性的，因此我們將在既有的 XOR 編碼技術的廣播排程方法上，再加上考量 Client 端請求具有時效性的因素，成為我們所提出的廣播排程方法，希望能降低客戶端的平均存取時間與提高滿足客戶端請求的數量。

第三節　XOR 編碼技術的介紹與運作

網路編碼（Network Coding）是最近幾年才發展出來的網路新技術，不但更新現有網路 store and forward 的架構，並且封包經過編碼再廣播的效能也比先前的一般廣播排程方法來的好，而網路編碼的核心概念是把多個封包，透過編碼技術結合成與原來大小一樣的一個封包，其編碼演算法只要是 Linear Algorithm 即可，Server 會將編碼封包廣播出去給所需要的 Client 端，不同的 Client 端收到編碼封包後，將其解碼便可以獲得各自所需要的資料或資訊。在網路編碼的過程中資料並完全沒有喪失，卻因為所要傳遞的封包數量減少，進而增進網路的效能。

XOR 運算是網路編碼常用的方法，透過圖 1-2 可以了解 XOR 運作方式。在圖 1-2(a)中，有四個資料項 d_1、d_2、d_3、d_4，其各自的編碼分別為 001、010、011、100。將資料項 d_1 與 d_3 和資料項 d_2 與 d_4，做 XOR 的編碼運算，運算過程與結果如圖 1-2(b)，其編碼後的結果分別為 010、110。假設我們今天想要獲得資料項 d_3 與 d_4，則必需要利用 Key 值 d_1 與 d_2 去解碼 010 與 110，進行 XOR 運算的動作，才能獲得所需要的資料項 d_3 與 d_4，如圖 1-2(c)。

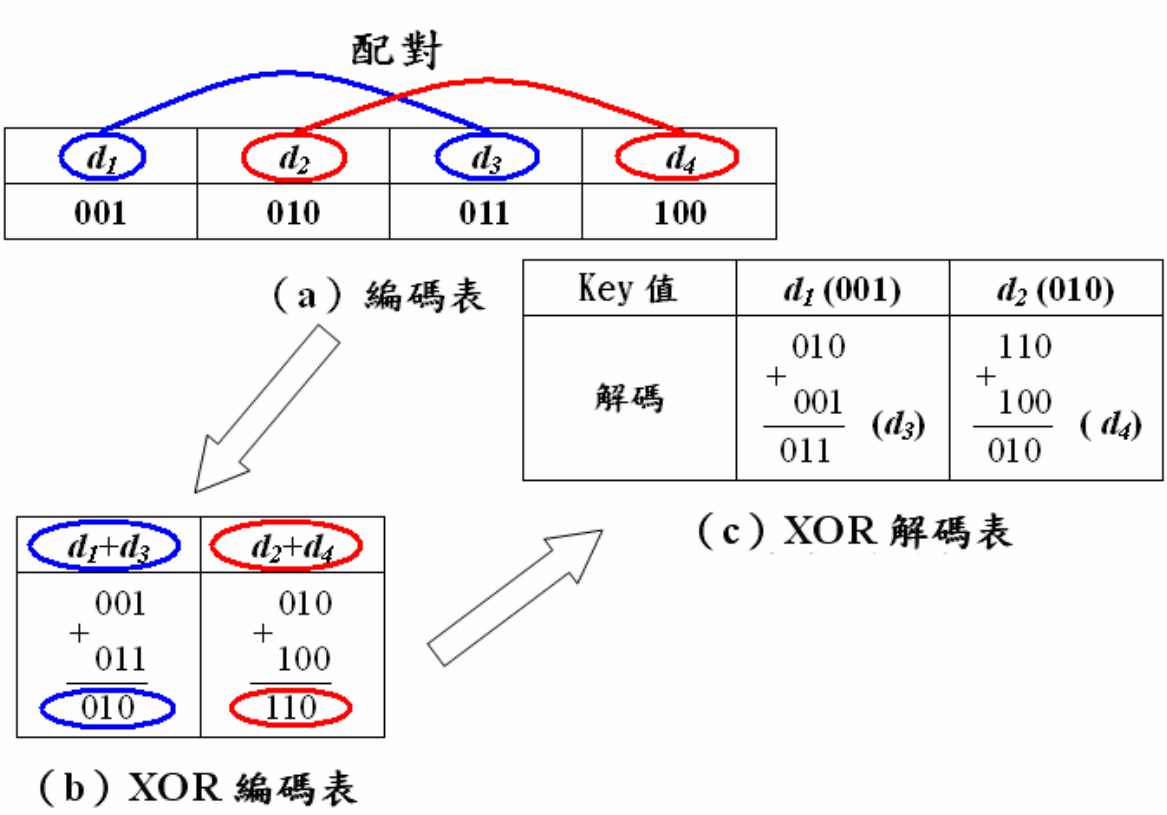

圖 1-2：XOR 實例運作圖

根據[3] [10]的研究，我們在此處介紹 XOR 編碼技術的實際運作過程。本研究假定 Client 端本身原本就有 Cache 一些資料項並且一個封包的長度等於一個廣播排程的時間單位。當 Client 端向 Server 發出 Query 之前，會先去 Cache Buffer 當中搜尋有無所需要的資料項，如果有則存取起來供使用者使用，如果沒有則對 Server 發出 Query。Client 端向 Server 端發出 Query 時，會將本身所 Cache 資料的資訊一併傳送給 Server，Server 端收到 Query 後，會分別建立兩張資料表以儲存 Query 與 Cache 的資訊。當 Server 選擇了要服務的 Query 時，Server 會將 Client 端 Query 與 Cache 的資料內容轉換成 0 與 1 所組成的機器語言（Machine Language），例如，

一個 Word 的文件檔最原始的結構為 0 與 1 的機器語言所組合而成的。Server 把 Query 與 Cache 的資料進行兩兩配對的 XOR 運算，將資料編碼壓縮在一起，便完成了資料的編碼組合，而 XOR 運算編碼後的資料長度與編碼前的長度是相同的。

資料編碼完成後，Server 便將編碼後的資料放入封包（Packet）中，該封包的標頭（Header）會包含編碼在一起的資料 ID、目的地與來源位址、編碼資料解碼後的檔案類型、大小或格式等資訊，最後將完整封包納入廣播排程當中進行廣播。

當編碼資料進行廣播時，Client 端會傾聽廣播排程裡封包的標頭是否有所需要資料，以及 Client 端本身是否有 Cache 相對應的解碼資料來進行解碼獲得所需要的資料，如果沒有則忽視，如果有就將其所需要的封包接收下來，由 Client 端本身所 Cache 的資料來進行解碼，Client 端的電腦會透過封包標頭裡所包含的資訊，將解碼後的資料轉換成所需要的資料，最後經由螢幕顯示給使用者觀看。

當 Client 端獲得所需要的資料後，會將該資料放到 Cache Buffer 中，以便下次有相同的需求時，可以直接在 Cache Buffer 中找到，而不用向 Server 端發出請求，可以降低等待時間。但是 Cache Buffer 大小並非是無限的，因此當 Cache Buffer 還有空位時，則將資料直接排入 Cache Buffer 中；如果 Cache Buffer 滿了，本研究是採用 Least Recently Used(LRU)的策略[17]，將 Cache Buffer 中，較沒有使用到的資料項從 Cache Buffer 中剔除，再將所要 Cache 的資料放入 Cache Buffer 中。

網路編碼的核心概念是將多個資料項編碼成一個長度大小相同的編碼資料項，雖然將越多個資料項編碼成一個編碼資料項越能節省頻寬的消耗，卻也造成了 Server 與 Client 編碼與解碼的運算複雜度提高、處理時間的拉長，連帶的使用者的等待時間也變長。例如，傳統的網路編碼技術把三個資料項編碼在一起並將資料項請求次數的多寡作為編碼時的係數$(2d_1+d_2+d_3)$、$(d_1+2d_2+d_3)$、$(d_1+d_2+2d_3)$，因此在進行解碼時需要存取兩個資料項來做聯立方程式進行解碼，因此在本研究我們採用兩個資料項為編碼的方式，使其在編碼與解碼的運作上較為簡單。

第四節　研究限制

本研究所提出的方法較貼近於真實的廣播環境，但是不適用於真實的廣播環境仍然有使用上的環境限制，而本研究的限制如下：

1. 廣播排程演算法適用於單頻道廣播系統，未考量到多個頻道的資料配置問題。
2. 適用於拉式或請求式的廣播系統（Pull or On-demand System）。
3. 固定長度的資料項，本研究中的資料項長度大小為 1。
4. 有限的無線網路頻寬。
5. 只考量到兩個資料項的編碼，未考量到多個資料項的編碼問題。
6. Client 端可以發出一個請求包含多個資料項，但是不能同時發送多個請求包含多個資料項。
7. Client 端本身假定已經事先 Cache 了一些資料項，作為編碼與解碼之用。
8. 未考量到 Server 從收到請求到資料項編碼、排程、廣播到 Client 端接收到編碼資料項所耗費的時間，本研究將這所耗費的時間大小設定為 0。

第五節　研究架構

本研究將以三個不同階段來循序進行介紹。第一階段，我們將先瞭解無線網路的背景環境及其運作的方式，接著進行相關文獻的探討。透過第一階段對無線網路有一定程度的瞭解後，擬定我們的研究方向和主題，並朝此研究方向進行探討。第二階段，我們將引用第一階段所獲得的相關知識和研究方向再針對內容進行問題討論，接著提出本研究專注的層面及欲解決的問題，接著探討可行方法。最後的一個階段是研究的驗證階段，在該階段我們將以程式來模擬無線廣播排程的實驗，檢定我們所提出方法是否能達到較佳的結果，並透過模擬實驗所獲得的數據分析比較，再提出本研究的結論。

透過上述三個研究步驟，可以完整的呈現我們的研究過程。本研究主要是針對在無線網路中，拉式廣播系統環境中頻寬不足及 Client 端發出的請求具有時效性的問題，提出一個合理且理想的解決方法。

第六節　簡介

在拉式廣播系統的架構下，我們發現目前所提出的廣播排程演算法大概有下列缺點：

1. 多數的排程演算法只是一味地降低 Client 端的等待時間，沒有考慮到在真實的廣播環境中，Client 端的請求通常是具有時效性的。
2. 以往的演算法大部分限制一個請求最多只能包含一個資料項，在真實的廣播環境中，是不合理的限制。
3. 先前的排程演算法大部分都專注在資料項的廣播排程上，進而在有限的頻寬下，盡可能滿足最多的使用者，少部分研究專注在有限的頻寬上，使其能容納更多使用者所需要的資料。

在真實的廣播環境中，Client 端發出的請求通常是具有時效性。假如只以客戶端發出請求的等待時間作為排程的依據，可能會導致某些時效性短、等待時間小的請求，會因為其本身等待時間小，而沒有安排在廣播排程上，反而是等待時間大，時效性過時的請求資料項優先納入廣播排程導致時效性短、等待時間小的請求超出其時效性而導致請求失敗並造成頻寬的浪費。因此，伺服器應該在每一個請求到達時，記錄其時效性並以此作為廣播排程的考慮因素。

另外，Client 端發出的請求通常是包含多個資料項並非只是單一個資料項而已，例如：Client 端請求要瀏覽一個網頁，而網頁是由數個元件所組成的，因此 Client 端就必須要收到所有請求的元件，才能開啟所要瀏覽的網頁。

在有限頻寬的情況下，先前大部分廣播排程的研究都以 Client 端請求資料項的冷熱門程度作為排程的考慮因素，使廣播排程滿足較多的使用者，但是所能滿足的使用者仍然有限。

為了解決上述所談論到的問題，我們採用[3]編碼技術的排程方法並加入考量請求的時效性來提高請求的成功率，因而提出了 On-demand with Deadline Encoding（ODE）演算法。在拉式廣播環境中， ODE 演算法的運作理念是透過網路編碼中 XOR 的編碼技術將兩個資料項結合編碼成一個編碼資料項並且配置在一個廣播週期的單位時間槽上進行廣播及利用本研究假設 Client 端已經事先 Cache 的一些資料項來進行解碼。ODE 演算法根據資料項的時效性、冷熱門程度，計算出其權重質最大的請求，再將該請求的 Client 端所 Cache 的資料項與請求的資料項進行兩兩的配對組合，接下來找出能滿足最多請求數量的資料項配對組合，進行資料項編碼 XOR 的運算動作將兩個資料項編碼成一個編碼資料項，並納入廣

播排程中，直到頻寬被塞滿為止，最後進行排程的廣播。由於編碼的技術，可以讓一個時間槽放入兩個資料項，也就是說一個時間槽可以一次服務多個請求，加上利用 Client 端本身所擁有的資料項作為解密的 Key 值，可以有效的減少頻寬的浪費並提高頻寬的使用，滿足更多的使用者所發出的請求。由於我們有考量到資料項的時效性、冷熱門程度，因此我們可以在時效性之內盡可能滿足最多的使用者。藉由上述的簡介可以得知，我們所提出的 ODE 演算法有著較佳的廣播效率。

本文將有系統的介紹其剩餘的部份：第二章將要探討在無線廣播環境下的相關文獻討論；第三章則是在描述我們在文獻探討所發現到的問題；第四章將詳細介紹 ODE 演算法的概念、程序及運作；第五章介紹我們的模擬實驗結果，並詳細說明我們的模擬環境、參數及數據分析；第六章是我們所得到的結論。

第二章、文獻探討

隨著資訊科技的快速進步，人們對於無線網路的應用已經越來越普遍。美國電機電子工程協會(Institute of Electrical and Electronics Engineers, IEEE）在 1990 年 11 月召開 802.11 的委員會；在 1997 年 6 月公佈 IEEE802.11 的共同標準協定，而在 IEEE802.11 的協定裡，規範了許多不同的需求與規格，使得 IEEE802.11 協定成為無線網路的另一個代名詞。

一般來說，無線區域網路（Wireless Local Area Network, WLAN），是將客戶端的請求經由機器設備的轉換，以電波的方式在媒介空氣中進行傳遞，送達伺服器端。伺服器端在接收到客戶端所發出的請求後，會依據其參考因素來進行資料項的廣播排程並以同樣的方式將資料項傳送給客戶端。簡單來說，就是客戶端透過無線網路設備，如：無線網路網卡（Wireless Card）與伺服器端的無線網路設備，如：無線基地台（Base Station, BS）或者是存取點（Access Point, AP）等，來進行連線的動作，從而可以交換資料項、連接上網際網路等。

另外一個無線廣域網路的普遍應用就是手機的普遍普及。在現今的社會環境中，手機幾乎是人手一機，甚至有一人多機的情況。實際上而言，手機通訊也是一種無線網路的應用，把手機視為是一個移動式的用戶端，在待機或者通訊時是透過無線電波與基地台進行連線的資料傳遞，但是一旦手機位於基地台的電波範圍之外就無法進行通訊的動作。

由上述所知，無線網路已經深深的融入到我們的生活當中。接下來本文將說明本研究的背景環境，並進一步討論相關的文獻。

第一節　背景環境

無線網路的主要架構可以區分成兩個部份：有基礎架構的無線網路（Infrastructure WLAN）和無基礎架構的無線網路（Noninfrastructure WLAN）；前者可以區分為拉式廣播系統（Pull System）與推式廣播系統（Push System），後者可以分為繞送表導向（Table Driven）和發送端要求導向（On-Demand）等不同的廣播環境，無線網路環境分類架構如圖 2-1。

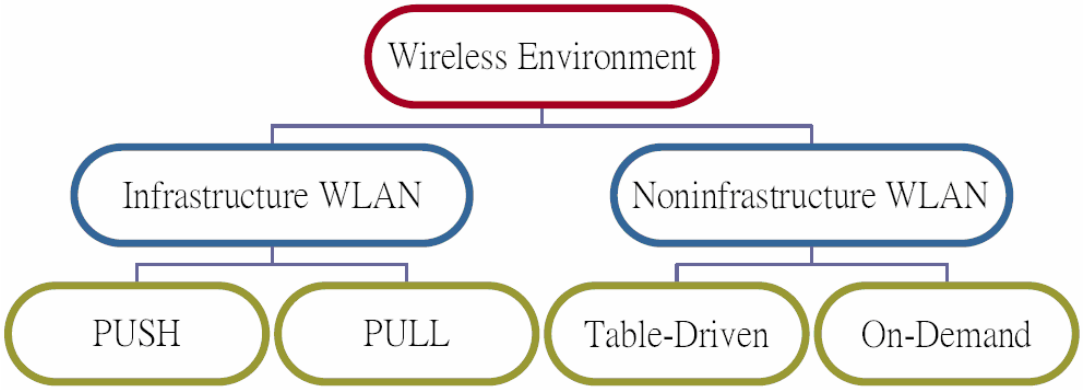

圖 2-1：無線網路架構分類圖

在有基礎架構的無線網路環境裡是以基地台為主，基地台和基地台之間透過無線網路來進行連接，但是在基地台廣播範圍內的移動式主機（Mobile Host, MH）可以直接與 Server 端進行連線。在該架構下，根據資料廣播排程的方式可以分成兩種：推式廣播和拉式廣播，也有結合推式和拉式的混合式廣播系統（Hybrid Broadcast System）。另外根據客戶端對資料項點播率的高低也就是客戶端對資料項的偏好程度，可以將資料項區分成兩類：點播率高的熱門資料項（Hot Data Item）與點播率低的冷門資料項（Cold Data Item）。

推式廣播也被稱為週期性廣播（Periodic Broadcast），推式廣播的運作方式是伺服器端不斷地週期性廣播所有的資料項，客戶端則是一直在監聽伺服器端所廣播的資料項，如果接聽到所需要的資料項時，再將資料項截取下來，其環境圖，如圖 2-2。

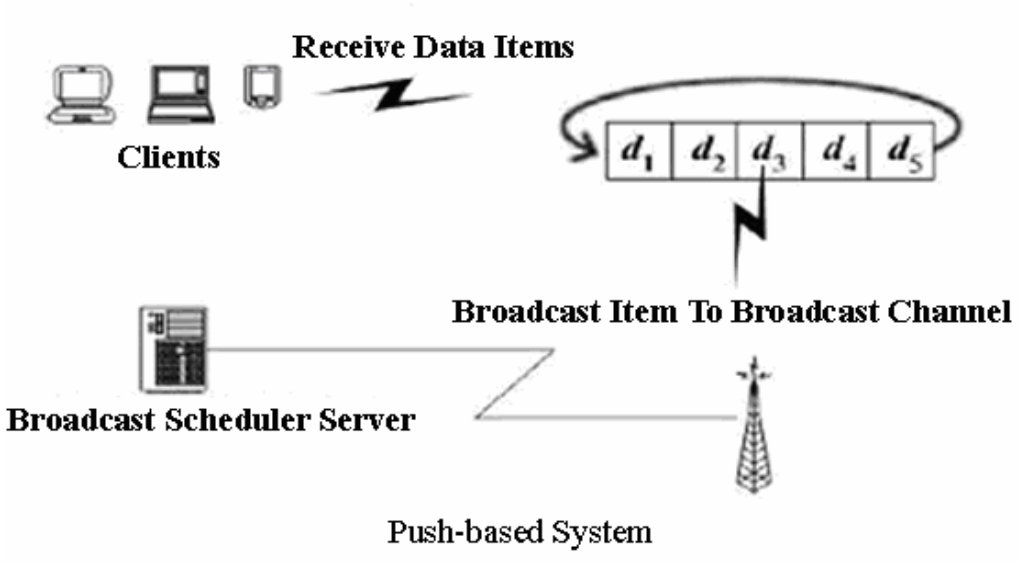

圖 2-2：推式廣播環境圖

拉式廣播也被稱為要求式廣播（On-Demand Broadcast），是由客戶端主動送出其需求資料項的請求廣播，經由上傳頻道傳送到 Server，Server 會依據客戶端對請求資料項的偏好程度、等待時間、時效性等等參考因素，利用一個較佳的廣播排程演算法，將客戶端所請求的資料項納入廣播排程當中，以滿足客戶端的請求，做最即時的資料項廣播排程，如圖 2-3。本文所提出的演算法屬於拉式資料廣播。

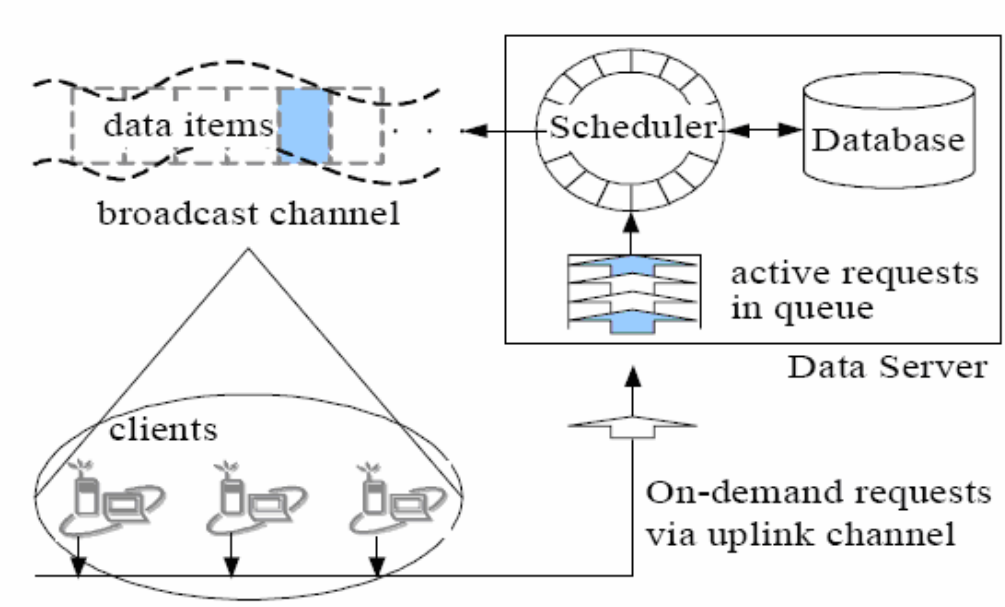

圖 2-3：拉式廣播環境圖

混合式廣播[6] [11]結合了推式的週期性廣播與拉式的有請求才廣播的特性，將頻寬分為主要頻道(Main Channel)與次要頻道(Sub Channel)。主要頻道採用推式的方式，將熱門的資料項放置於廣播排程，週期性的廣播熱門資料項，以滿足絕大部分的使用者。次要頻道平常是處於閒置的狀態，一旦有使用者發出請求的資料項不在主要頻道的廣播排程上，也就是所謂的冷門資料項，該請求則透過次要頻道向伺服器請求資料項，伺服器端再經由次要頻道將資料項傳遞至客戶端。混合式廣播的議題主要專注在主要與次要頻道的頻寬配置與資料項的廣播排程，進而有效率的利用頻寬來滿足最多的使用者，如圖 2-4。

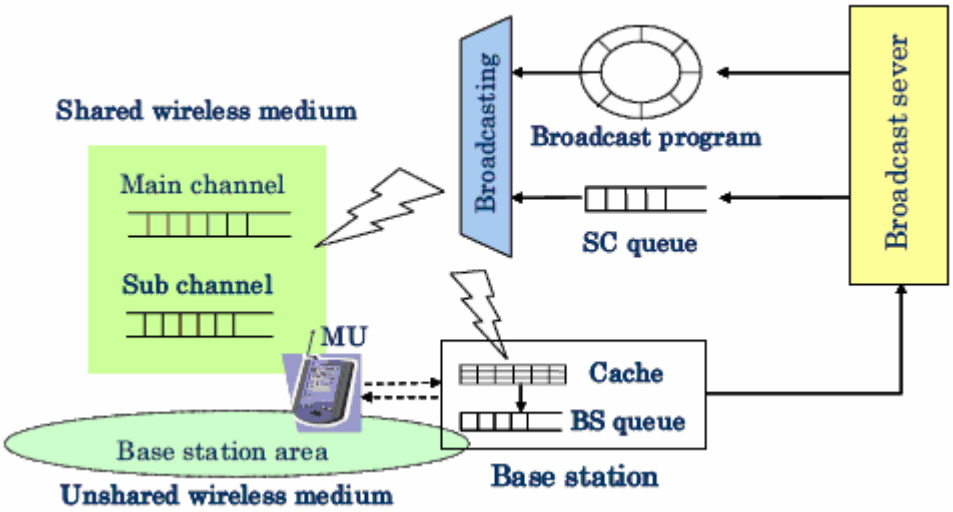

圖 2-4：混合式廣播環境圖

第二節　相關文獻

在非對稱式通訊環境中的推式廣播方法，假設廣播頻寬大小與資料項長度是固定的，資料項是以轉盤的方式來排程並週期性的廣播，雖然在廣播排程上，所有的資料項都可以接收的到，但是並沒有考量到客戶端發出請求的樣式(Pattern)、資料項的偏好程度等因素，因為在真實的廣播環境中，資料項的點播率有冷熱門之分，而推式廣播的每個資料項

廣播頻率都是一樣的，假設有個熱門資料項，而請求該資料項的大多數使用者錯過該資料項的廣播時間時，則必須等待所有資料項都廣播循環一次之後，才能接收到其所需要的熱門資料項，而等待時間往往過長導致請求失敗；另外有些冷門的資料項可能根本沒有使用者請求，由於推式廣播的特性也將其安排置於廣播排程上，造成頻寬的浪費。

在假定廣播頻寬大小與資料項長度固定的要求式廣播環境下，其產生一個有效率的廣播排程，通常會考量到兩種問題：第一種是選擇請求的問題(Query Selection Problem)、第二種是廣播排程中資料項配置的問題(Broadcast Scheduling Problem)。

選擇請求問題是當 Client 端向 Server 端發出請求時，Server 將這些請求 Cache 在佇列中等待服務，而要依據什麼來選擇這些還沒有被服務的請求將其納入排程，於是產生選擇請求的問題。在此依序介紹在三種不同假設條件下，由先前的研究所提出來的演算法。

一、 在要求式廣播系統下，單一資料項請求的廣播排程：

主要是根據客戶端請求的樣式、等待時間、時效性及資料項的冷熱門程度等等，作為資料項廣播排程的依據，以下將分別介紹 First Come First Service(FCFS)、考量客戶端請求等待時間的演算法 Longest Wait First (LWF)[16]、考量請求時效性的演算法 Earliest Deadline First (EDF)[15]、考量資料項的冷熱門程度的演算法 Most Requests First(MRF)、R×W[7]。

FCFS 是依照請求到達 Server 的先後順序來決定廣播排程的優先順序，最先到達的最先廣播，以此類推。在資料項的冷熱門程度差異小的廣播環境中，FCFS 演算法相較於其他一個請求只包含一個資料項的演算法的存取時間要來的小，反之 FCFS 演算法的存取時間相較於其他演算法較大，而且 FCFS 演算法運用在單一請求只包含單一資料項且資料項的冷

熱門程度大或小的廣播環境中，Client 端存取資料項的時間幾乎是相同的，因此 FCFS 演算法不受請求頻率和請求具有時效性的影響。另外在一個請求包含多個資料項的廣播環境中，FCFS 相較於其他單一個請求只包含一個資料項的排程演算法的 Client 端存取時間較小。

MRF 從還沒有被服務的請求當中，選出請求頻率最高的資料項優先廣播；請求頻率指的是資料項被請求的次數，請求的次數越少代表該資料項越冷門，反之則越熱門。MRF 演算法是根據資料項的冷熱門程度來做排程，因此當資料項的冷熱門程度差異大時，MRF 被證明能減少客戶端的等待時間並滿足較多的請求。但是如果將 MRF 演算法運用在多個資料項請求的廣播環境中，單一請求所包含的多個資料項的冷熱門程度可能是不相同的，因此在廣播排程的位置上可能是不連續，這將會拉長客戶端的存取時間，更糟的是萬一該請求所包含的多個資料項當中包含冷門的資料項，該請求會一直等不到所需要的資料項而產生飢渴的現象，最後可能因為等待時間過長而超過時效性導致請求失敗。

由 J. W. Wong.所提出的 LWF 演算法，主要是以縮短客戶端請求的等待時間為主要訴求，因此將客戶端請求相同資料項的請求的等待時間進行加總，加總後等待時間最長的資料項最優先納入廣播排程。LWF 將有請求相同資料項請求的等待時間進行加總，因此 LWF 演算法有間接的考量到請求的頻率，所以在資料項冷熱門程度差異大的要求式廣播環境中，LWF 演算法仍然能有效降低 Client 端的等待時間與滿足較多的請求。

D. Aksoy 等人發現了客戶端請求的資料項有冷熱門之分，因此將資料項的請求比率也就是冷熱門程度納入廣播排程的參考因素並延續了 J. W. Wong.提出的 LWF 演算法中，考量等待時間的觀念，而提出了 R×W 演算法。R×W 演算法中的 R 代表該資料項被尚未被服務的請求所請求的

頻率也就是該資料項的冷熱門程度、W 是代表尚未被服務的請求的等待時間；R×W 演算法將 R 與 W 做相乘的動作，所獲得的乘積也就是權重值，做為資料項廣播排程時的依據，其權重值越大者越優先廣播。R×W 演算法結合了 MRF 與 LWF 演算法的優點，在資料項的請求頻率與請求的等待時間取得一個平衡，一方面考量資料項的冷熱門程度是為了能夠滿足較多的請求；另一方面考量請求的等待時間可以避免因為等待時間過長而造成飢餓（Starve）的情況發生。R×W 雖然能滿足絕大部分使用者的請求並減少飢餓的情況發生，但是仍然沒有將客戶端請求的時效性納入廣播排程的依據，因此時效性較為急迫的請求，可能會因為時效性的過期，導致請求的失敗。

考量到請求時效性的 EDF 演算法是由 P. Xuan 等人所提出，EDF 演算法是以客戶端請求的時效性作為廣播排程的依據，時效性越短的越優先廣播，因此能有效減少客戶端的請求超過時效性，而導致的請求失敗率。雖然 EDF 演算法能有效地降低請求的失敗率，但是僅以時效性作為廣播排程的依據，有可能導致無法滿足絕大部分使用者的請求，舉例來說，假設客戶端請求的資料項，其時效性最短但是卻是屬於冷門的資料項，根據 EDF 演算法將會優先廣播，因此絕大部分請求熱門資料項的客戶端都處於饑餓的狀態，並且至少必須等待一次的廣播排程才有可能接收到其所需要的熱門資料項能，而等待的時間越長越有可能超出其本身的時限，而導致請求的失敗，同時也無法滿足絕大部分使用者的請求。

二、　在要求式廣播系統下，多資料項請求的廣播排程：

在真實的無線網路廣播環境中，Client 端發出的請求通常是包含多個資料項，因此將上述一個請求只包含一個資料項的廣播排程演算法套用在多個資料項請求的廣播環境中，將會導致存取時間的拉長並且提高飢

餓與請求失敗的情況產生。以下將介紹在多資料項請求環境中的排程演算法 QEM(Query Expansion Method)[9]。

QEM 是以 Query 的冷熱門程度來決定排程的先後順序，QEM 排程的方法，首先將請求從熱門排到冷門並依序展開所包含的資料項。在展開的過程當中，先前已經展開的 Query 必須保持原本的長度並調整資料項的位置，讓有相依性的資料項可以更緊密的排在一起，透過這樣的運作方式可以減少 Client 端的存取等待時間。

三、　在要求式廣播系統下，運用編碼技術的多資料項請求的排程：

根據上述，我們得知不管是推式廣播還是以考量等待時間、時效性、資料項的冷熱門程度等為廣播排程依據的單一資料項請求的拉式廣播或是多資料項請求的拉式廣播，主要都是為了能有效地利用有限的頻寬，來滿足最多的使用者並降低 Client 端的存取時間，然而再怎麼優秀廣播排程演算法，在有限的頻寬下，所能滿足的使用者仍然有限，因此有研究[3] [10]提出結合網路編碼的技術使其有限的頻寬可以相較於以往容納更多的資料項並且服務更多的請求，進而滿足更多的使用者。

先前的研究[14] 指出網路編碼技術可以有效地利用有限的頻寬去做多個資料的傳輸通訊，具體來說就是透過伺服器將多個資料項，以線性組合的方式編碼產生單一個編碼資料項，再將編碼資料項與解密的 key 值一起傳送到客戶端，再由客戶端利用所收到的 key 值來進行編碼資料項的解碼，獲得客戶端本身所需要的資料項，然而在傳送 key 值到客戶端時，會消耗其有限的頻寬。

在[3]中所提出的 OE 演算法與先前在單一資料項請求或是多資料項請求環境下的相關研究的最大不同在於以往的演算法一個廣播時間單位只能放置一個資料項，OE 則是將兩個不同的資料項透過 XOR 運算結合

成一個編碼資料項，再放入排程當中，因此 OE 的方法可以在一個時間槽（Time Slot）服務兩個不同資料項的請求並且可以減少頻寬的消耗，圖 2-5 為一般廣播排程與 OE 演算法的差別。

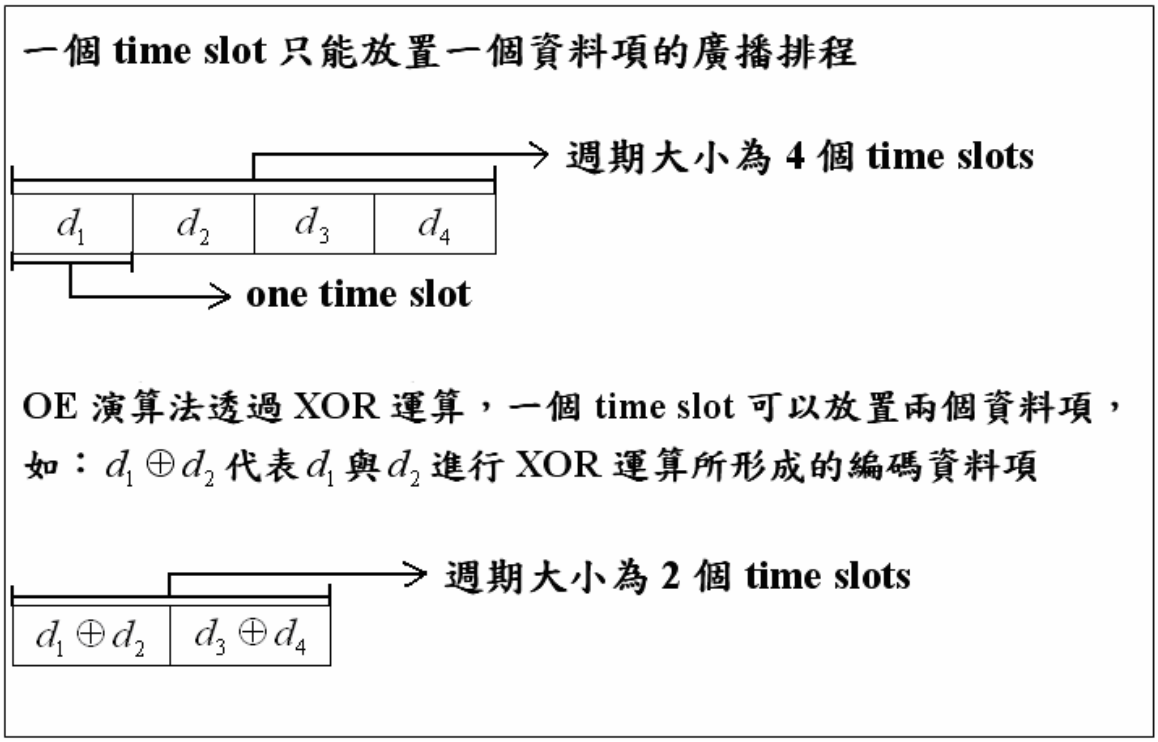

圖 2-5：一般廣播排程與 OE 演算法的差別

OE 演算法將網路編碼技術應用於要求式資料廣播環境中，將請求資料項的冷熱門程度與請求的等待時間作為排程的考慮因素，並且運用假定客戶端本身已經事先 Cache 的一些資料項，作為編碼資料項解碼時的 key 值使用，相較於先前所提到的 FCFS、LWF、MRF、EDF、R×W，OE 演算法不僅打破了一個廣播時間單位只能存放一個資料項的限制，並且改良了傳統網路編碼必須將 key 值與編碼資料項一併傳送到客戶端造成頻寬消耗的缺點，因此 OE 演算法有效的降低資料項的平均存取時間並且減少頻寬的消耗，同時滿足了較多的使用者所送出的請求。在圖 2-6 中，OE 演算法相較於一般廣播排程，頻寬只消耗 4 個 time slot，另外由於 OE

改良了傳統網路編碼必須將 key 值與編碼資料項一併傳送到客戶端所造成頻寬消耗的缺點，因此 OE 演算法比傳統的網路編碼節省了 1 個 time slot 的頻寬消耗。

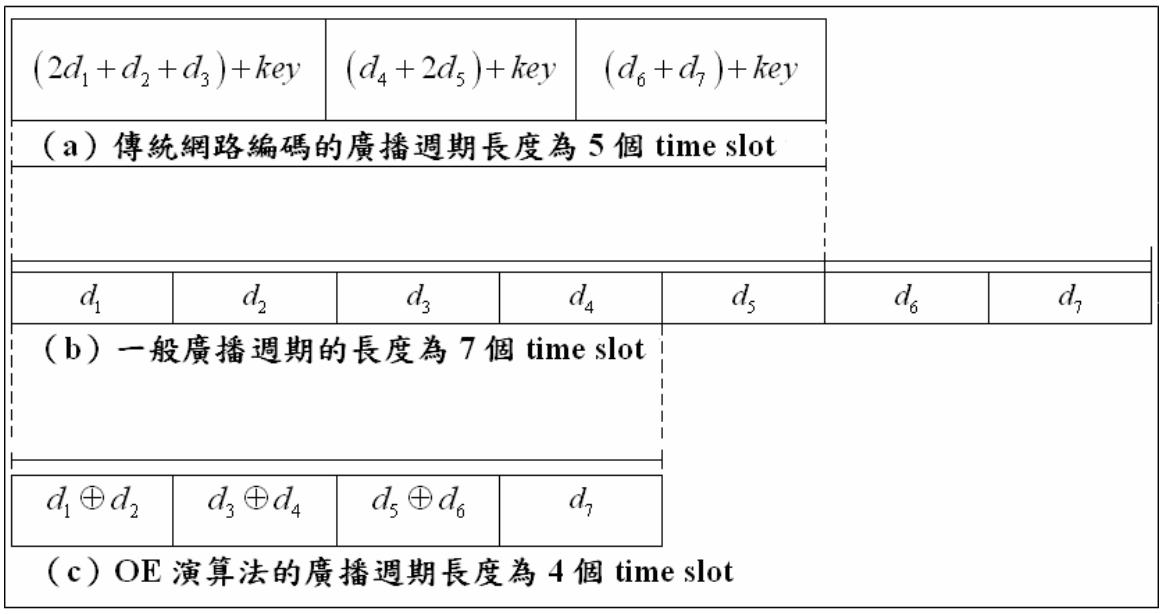

圖 2-6：比較頻寬的消耗

在此介紹 OE 演算法的運作過程，OE 的運作過程大概可以分成三個步驟：選取尚未被服務的請求、請求所包含的資料項的編碼與排程、廣播。

第一步驟，Server 會從尚未被服務的請求中，選取優先服務的請求，決定請求被服務的先後順序是以該請求的資料項出現在所有尚未被服務請求中的平均次數乘上其 Client 端本身所 Cache 的資料項在所有 Cache 中平均出現的次數再乘以請求的等待時間，所獲得的乘積就是衡量請求被選取的權重值，權重值越大就越優先被選取。

第二步驟為編碼與排程，選取所要優先排程的請求後，將請求所包含資料項與其 Client 端所 Cache 的資料項進行兩兩配對的編碼組合，再將編碼後的編碼資料項所能服務的請求數量去乘以所服務的請求的等待時間的總和，所得到的乘積就是該編碼資料項的權重值，權重值越大者

越優先納入排程。

第三步驟，Server 將排程廣播出去，並且刪除佇列中已經被服務完成的請求以及更新這段時間到達 Server 端的請求和尚未被服務完成的請求與其 Client 端所 Cache 的資料項，反覆進行步驟一到步驟三的動作，直到所有的請求都被滿足為止。

上述大致上說明了 OE 演算法的運作過程，以下將透過實際的例子，來詳加說明，如圖 2-7。

在圖 2-7(a)中，有四個請求各自代表不同的 Client 端，包含了本身所 Cache 的資料項、請求的資料項、waiting time、deadline 的資訊。

根據上述的 OE 演算法的運作過程，首先 Server 會先選取所要服務的請求，而決定服務的優先順序是以 p_i 值的大小來決定，p_i 值代表權重其權重值越大者越優先服務，p_i 值的計算方式是 $P_i = Req(Q_i) \times Store(S_i) \times w_i$，$Req(Q_i)$ 是請求 Q_i 的資料項出現在所有尚未被服務請求中的平均次數，$Store(S_i)$ 是 Client 端本身所 Cache 的資料項在所有 Cache 中平均出現的次數，w_i 是請求 Q_i 的等待時間，如圖 2-7(b)，Q_1 的 p_i 值最大，優先服務。

選擇所要服務的請求後，便將該請求的請求資料項與 Cache 的資料項兩兩做編碼配對，並計算編碼的強度 $DI(e_j)$，強度越強越優先納入排程，計算方式為 $DI(e_j) = |Dem(e_j)| \times \sum_{u_i \in Dem(e_j)} w_i$，$|Dem(e_j)|$ 是編碼資料項 e_j 所能服務的請求數量，$\sum_{u_i \in Dem(e_j)} w_i$ 是編碼資料項 e_j 所服務的請求的等待時間的總和，如圖 2-7(c)。

接著將編碼資料項依照強度由高到低，納入排程直到沒有空位，如圖 2-7(d)，編碼資料項 $d_1 \oplus d_2$ 與 $d_4 \oplus d_5$ 被納入排程。假如 Q_1 的編碼資料項已全部排入廣播排程，而排程仍有空位則由次高 p_i 值的 Q_4，將請求的資

料項與 Cache 的資料項兩兩做編碼配對，並計算編碼的強度 $DI(e_j)$，強度越強越優先納入排程，反覆進行上述第一步驟與第二步驟，直到排程沒有空位，再進行廣播。

第一次廣播結束後，請求 Q_1、Q_3、Q_4 尚未被滿足並且又有新的請求到來，如圖 2-7(e)。根據步驟一，計算 p_i 值並選出所要服務的請求，如圖 2-7(f)。選擇所要服務的請求後，便將該請求的請求資料項與 Cache 的資料項兩兩做編碼配對，並計算編碼的強度 $DI(e_j)$，如圖 2-7(g)。編碼資料項的強度越強越優先納入排程，直到排程沒有空位並廣播，如圖 2-7(h)。在第二次廣播結束後，沒有新的請求產生同時也滿足了所有的請求，因此中止 OE 演算法的執行。

Request	Storage	Request	Waiting time (time slot)	Deadline (time slots)
Q_1	$S_1=\{d_2,d_4\}$	$Q_1=\{d_1,d_5,d_6\}$	2	5
Q_2	$S_2=\{d_1,d_3,d_4\}$	$Q_2=\{d_2,d_5\}$	1	2
Q_3	$S_3=\{d_1,d_4,d_5\}$	$Q_3=\{d_2,d_3\}$	1	1
Q_4	$S_4=\{d_2,d_5\}$	$Q_4=\{d_1,d_3\}$	2	1

(a)客戶端發出請求的資料項、客戶端儲存的資料項、waiting time

Request	$Store(S_i)$	$Req(Q_i)$	p_i
Q_1	5/4	5/4	25/8
Q_2	3/2	1	3/2
Q_3	7/4	1	7/4
Q_4	1	1	2

(b)計算每個請求的 p_i 值

Encoding data e_j	$Dem(e_j)$	Total waiting time	$DI(e_j)$
$d_1 \oplus d_2$	$\{u_1,u_2,u_3,u_4\}$	6	24
$d_1 \oplus d_4$	$\{u_1\}$	2	2
$d_2 \oplus d_5$	$\{u_1,u_3\}$	3	6
$d_4 \oplus d_5$	$\{u_1,u_2\}$	3	6
$d_2 \oplus d_6$	$\{u_1\}$	2	2
$d_4 \oplus d_6$	$\{u_1\}$	2	2

(c)取 $DI(e_j)$ 最大值的優先納入排程

$d_1 \oplus d_2$	$d_4 \oplus d_5$

(d)第一次廣播排程

圖 2-7：OE 演算法實例說明

Request	Storage	Request	Waiting time (time slot)	Deadline (time slots)
Q_1	$S_1=\{d_1,d_2,d_4,d_5\}$	$Q_1=\{d_6\}$	**4**	**3**
Q_3	$S_3=\{d_1,d_2,d_4,d_5\}$	$Q_3=\{d_3\}$	**3**	**-1**
Q_4	$S_4=\{d_1,d_2,d_5\}$	$Q_4=\{d_3\}$	**4**	**-1**
Q_5	$S_5=\{d_1,d_3\}$	$Q_5=\{d_5,d_6\}$	**8**	**4**

(e)新的 request 產生與尚未被服務完成的 request

Request	$Store(S_i)$	$Req(Q_i)$	p_i
Q_1	3	1/2	6
Q_3	3	3/4	27/4
Q_4	5/2	3/4	15/2
Q_5	5/4	1	10

(f)計算每個請求的 p_i 值

Encoding data e_j	$Dem(e_j)$	Total waiting time	$DI(e_j)$
$d_1 \oplus d_5$	$\{u_5\}$	**8**	**8**
$d_1 \oplus d_6$	$\{u_1,u_5\}$	**12**	**24**
$d_3 \oplus d_5$	$\{u_3,u_4,u_5\}$	**15**	**45**
$d_3 \oplus d_6$	$\{u_5\}$	**8**	**8**

(g)取 $DI(e_j)$ 最大值的優先納入排程

$d_3 \oplus d_5$	$d_1 \oplus d_6$

(h)第二次廣播排程，滿足了所有的請求

圖 2-7：OE 演算法實例說明

介紹完了關於選擇請求的問題(Query Selection Problem)之後，接下來將介紹排程中資料配置的問題(Broadcast scheduling Problem)。排程中資料項配置的方式有兩種：第一種是在同一個廣播週期中，同一個資料項只有配置一次並沒有重複出現；另一種是在同一個廣播週期中，資料項可以重複配置，如圖 2-8。

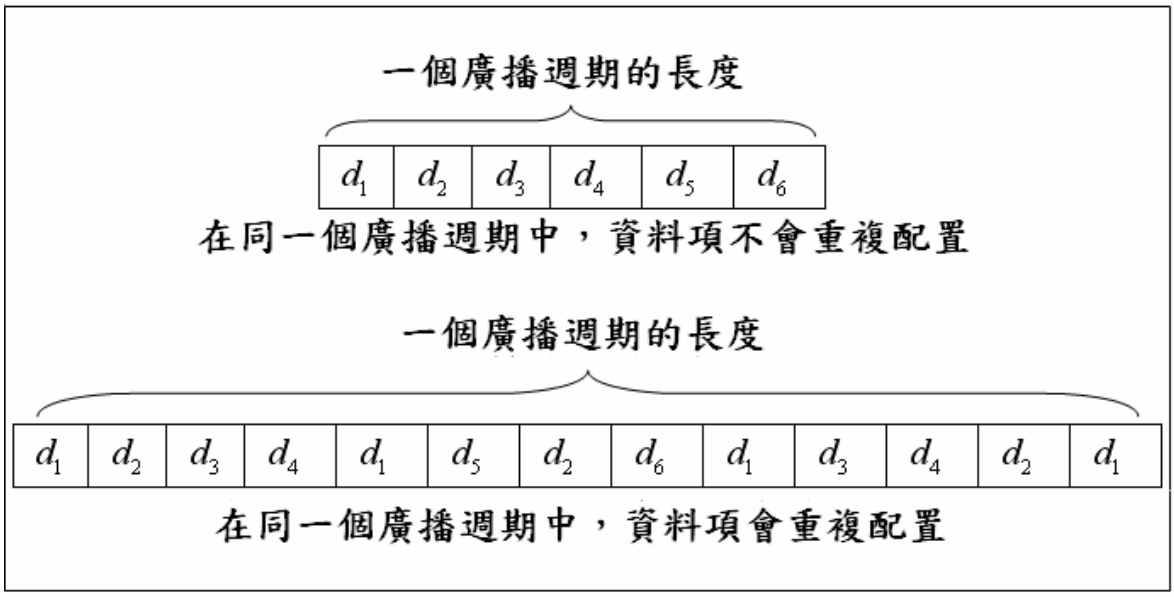

圖 2-8：資料項在排程中的配置方式

先前的研究[3] [7] [9] [10] [11] [13] [15][16]採用的方式都是在同一個廣播週期資料項不可以重複配置，也就是說在一個廣播週期中，不會有重複的資料項出現，這種方法的廣播週期長度較短也比較節省頻寬，因此下一個廣播週期可以較快被廣播，但是一旦 Client 端錯過現行廣播週期中所需要的資料項，則必須等待下一個廣播週期才能收到所需要的資料項。

研究[4][5][6][12]中的資料配置方式是在同一個廣播週期中，資料項可以重複配置，優點是 Client 端一旦錯過所需要的資料項，還有機會再從該資料項重複出現時候，進行接收；缺點是重複配置資料項的廣播週期較長，因此下一個廣播週期必須要等待較久的時間才能進行廣播。常

見的資料項重複配置方式是根據資料項本身的冷熱門程度來決定擺放的頻率，如圖 2-8，資料項d_1、d_2、d_3、d_4都是熱門資料項，在廣播週期中出現的頻率依序為 3、4、5、6，在圖 2-8 中的擺放頻率指的是兩個相同的資料項間，間隔幾個 slot，假如要放置的位置已經有排放資料項，則順移到下一個空的 slot，進行排放。等到所有熱門資料都已經排放完畢後，如有空的 slot 則擺放冷門的資料項將排程排滿，最後進行廣播。

但是 OE 演算法在考量客戶端請求的部份，主要是以請求的等待時間來做為考量的依據，並不符合真實廣播環境中，請求通常具有時效性的特性。

根據上述各個演算法所面臨到的問題，我們提出了 On-demand with Deadline Encoding（ODE）演算法來解決上述問題，ODE 演算法主要是延續了 OE 演算法的想法與概念，改良了 OE 演算法，將廣播排程依據的客戶端請求的等待時間改為以客戶端請求的時效性，使其能夠更加的貼近於真實的廣播環境。在第三章我們會更詳盡地描述以往演算法所面臨到的問題。

第三章、問題描述

在前一章節的部份，我們已經清楚地介紹一些相關的文獻著作。接著本章節將會詳細說明在文獻探討中，所發現的問題。本章節將有系統的介紹衡量尺度、符號的定義與說明、系統環境架構、編碼與時效性的問題。

第一節　衡量尺度

在本文中，用來評估廣播排程效能好壞的指標有平均存取時間（Average Access Time, AAT）和請求的成功率（Success Rate, SR）。存取時間（Access Time, AT）是指從 Client 端發出請求後，到 Client 端完全收到所需要的資料項的這段時間，如圖 3-1，Q_i 請求了三個資料項 d_1 、d_2 、d_5 ，因此 Q_i 必須要接收到 d_5 才能滿足，而這段時間就是存取時間。

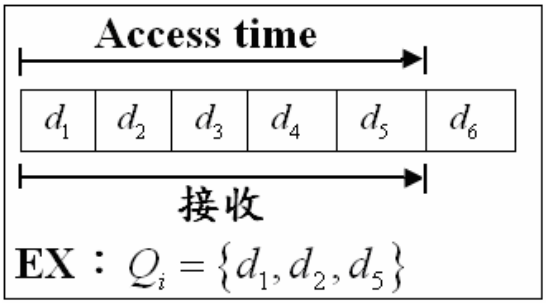

圖 3-1：存取時間的例子

我們做廣播排程的目的就是要找出一個理想的方法可以降低 Client 端的平均存取時間與提高請求的成功率。計算平均存取時間的方程式可以表示為公式（1），將所有被滿足的請求的存取時間總和除以所有到達 Server 的請求數量；成功率的方程式可以表示為公式（2），所有被滿足的請求數量除以所有到達 Server 的請求數量。

$$AAT = \frac{Total\ access\ time\ of\ \ arrival\ requests}{Number\ of\ \ arrival\ requests} \qquad (1)$$

$$SR = \frac{Total\ number\ of\ \ successful\ requests}{Number\ of\ \ arrival\ requests} \qquad (2)$$

第二節　符號的定義與說明

表 2 定義本研究使用的標記符號，以下說明本研究用到的標記符號的意義。

表 3-1：符號定義表

符號	說明
Q_i	user i 所發出的請求
Q	所有請求的集合
$\lvert Q \rvert$	請求的數量
t_i	Q_i 的 deadline
S_i	user i 所 cache 的資料項集合
e_j	第 j 個編碼資料項
D	資料庫裡資料項的集合
$\lvert D \rvert$	資料庫裡資料項的數量
$Dem(e_j)$	編碼資料項 e_j 可以服務多少個請求
$DE(e_j)$	編碼資料項 e_j 時效性急迫程度的權重
n_u	發出請求的使用者數量
d_j	資料庫裡第 j 個資料項
L	一個廣播週期長度
u_i	user i
$Store(S_i)$	S_i 所包含的資料項在所有 cache 裡平均出現的次數
$Req(Q_i)$	Q_i 所包含的資料項在所有請求裡平均出現的次數
s_{dj}	d_j 資料項在所有 cache 裡出現的次數
r_{dj}	d_j 資料項在所有請求裡出現的次數
p_i	決定選擇哪一個請求，來進行編碼配對服務的權重

在表 3-1 中，D指的是資料庫中所有資料項的集合，D可以表達成$D=\{d_1,d_2,d_3,\ldots,d_n\}$，$d_j$指的是資料庫中第 j 個資料項，$|D|$是資料庫中所有資料項的數量。Q_i代表的是第i個使用者所發出的請求，$|Q|$是所有請求的數量，t_i是第i個使用者所發出的請求Q_i的 deadline。第i個使用者的客戶端所存取資料項的集合也就是 Cache 則標示為S_i。

r_{d_j}標記代表資料項d_j在所有請求的資料項集合Q中出現的次數，而$Req(Q_i)$則代表這個請求Q_i的資料項在所有請求裡出現的平均次數，其計算方式為公式（3）。

$$Req(Q_i)=\frac{\sum_{d_j\in Q_i} r_{d_j}}{|Q|} \qquad (3)$$

s_{d_j}標記代表資料項d_j在所有 Cache 的資料項集合中出現的次數，而$Store(S_i)$則代表這個 Cache 所包含的資料項S_i在所有 Cache 裡出現的平均次數，第i個使用者的客戶端所存取資料項S_i這個 Cache 的資料項在所有 Cache 裡出現的平均次數，計算方式為公式（4）。

$$Store(S_i)=\frac{\sum_{d_j\in S_i} s_{d_j}}{|Q|} \qquad (4)$$

在圖 3-2 中，透過實際的例子來了解公式（3）、（4）的運算。

Request	Storage	Request
Q_1	$S_1=\{d_2,d_4\}$	$Q_1=\{d_1,d_5,d_6\}$
Q_2	$S_2=\{d_1,d_3,d_4\}$	$Q_2=\{d_2,d_5\}$

$Store(S_1)=1.5$，其運算過程如下：

$$Store(S_i)=\frac{\sum_{d_j\in S_i} s_{d_j}}{|Q|}=\frac{s_{d_2}+s_{d_4}}{|Q|}=\frac{1+2}{2}=1.5$$

$Req(Q_1)=2$，其運算過程如下：

$$Req(Q_i)=\frac{\sum_{d_j\in Q_i} r_{d_j}}{|Q|}=\frac{r_{d_1}+r_{d_5}+r_{d_6}}{|Q|}=\frac{1+2+1}{2}=2$$

圖 3-2：舉例說明公式(3)、(4)

當演算法在運作時，Server 端會根據 p_i 值的大小來決定優先服務哪一個請求，p_i 值越大的越優先廣播，p_i 的計算方式為公式（5）。

$$p_i=Req(Q_i)\times Store(S_i)\times\frac{1}{t_i} \qquad (5)$$

e_j 代表 Server 端將選取的 Request 與 Cache 的資料項，兩兩配對組合而成的編碼資料項。例如：Server 選擇了圖 3-2 中的 Q_1 為所要服務的請求，因此將 $S_1=\{d_2,d_4\}$，$Q_1=\{d_1,d_5,d_6\}$ 做編碼配對產生 6 個編碼資料項，$d_1\oplus d_2$、$d_1\oplus d_4$、$d_2\oplus d_5$、$d_4\oplus d_5$、$d_2\oplus d_6$、$d_4\oplus d_6$。

編碼資料項產生後，Server 會根據 $DE(e_j)$ 值的大小來決定是否放入排

程的依據，$DE(e_j)$值越大的越優先廣播，$DE(e_j)$的計算方式為公式（6）。$|Dem(e_j)|$代表編碼資料項e_j可以服務多少個請求的數量。

$$DE(e_j)=|Dem(e_j)|\times\left(\sum_{u_i\in Dem(e_j)}\frac{1}{t_i}\right) \qquad (6)$$

第三節　系統環境架構

本研究主要探討的是在多資料項請求的要求式廣播系統下，採用編碼技術的排程方法，其架構如圖 3-3。

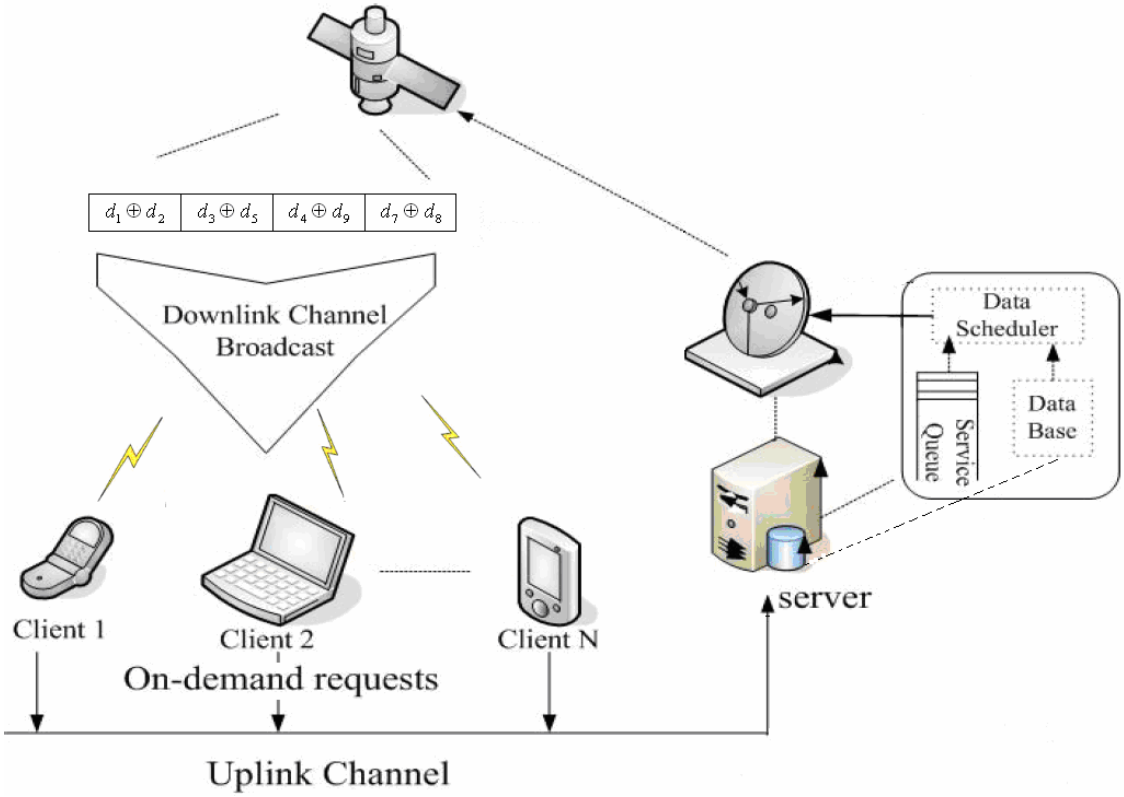

圖 3-3：採用編碼技術的系統架構圖

要求式廣播系統是根據 Client 端的請求 Server 再給予回應，因此 Client 端會經由上傳頻道發送請求與 Client 端本身所 Cache 資料項的資訊給 Server，Server 接收到請求後，會先存到佇列中並等待服務。隨著演算法技巧與考慮因素的不同，資料項納入排程的先後順序也不相同。

本研究採用編碼的技術，所以排程中的資料項是編碼後的編碼資料項，其編碼的方式是將 Request 與 Cache 的資料項轉成 0 與 1 的資訊，再互相兩兩配對組合成編碼資料項納入排程當中。排程產生後，Server 就將排程經由 Downlink channel 廣播出去，Client 端此時則在傾聽封包標頭所含的資訊是否有所需要的資料項，如果沒有就等待下一個廣播排程，如果有需求的資料項就將編碼資料項接收下來，並用封包標頭裡的解碼資訊與 Client 端本身所事先 Cache 的資料項來進行解碼，獲得所需要的資料項。以上為本研究的系統架構的運作， Request 與 Cache 的資料項的編碼與解碼的運算方式是採用 XOR 的運算方法，而 XOR 編碼的運算方式已經在第一章的第三小節有了詳細的介紹。

第四節　時效性與編碼的問題

在文獻探討的部份，我們了解到各式各樣的演算法，大部分主要訴求是降低 Client 端的平均存取時間、提高請求成功的比率與盡可能的滿足最多的請求數量。

在降低 Client 端存取時間的相關研究上[3] [4] [7] [9] [12]，大都以考量 Client 端請求的等待時間大小，來做為資料項的排程並廣播的依據，然而在真實的廣播環境中，Client 端發出的請求通常是具有時效性的，一旦請求的等待時間超過本身的 deadline 時，將會導致請求的失敗，因此在資料項的排程編排時，應該以請求的時效性來做為排程的參考因素。透過圖 3-4 所舉的例子來說明，為什麼要以請求的時效性來做為排程的參考因素。在圖 3-4（a）中，S 代表伺服器，C_1、C_2、C_3、C_4 分別代表四個不同的客戶端，其本身分別各自擁有資料項 d_3、d_2、d_1、d_3，各自請求的資料項有 d_1、d_3、d_3、d_2，廣播一個資料項所耗費的時間為 1 秒。圖 3-4（b）是根據請求的等待時間的大小，作為廣播排程的參考因素，

等待時間越大的請求越優先廣播，然而 C_4 請求資料項 d_2 時，必須要等 2 秒才能接收到 d_2，因此超過 C_4 請求的時效性，所以 C_4 的請求失敗。圖 3-4（c）將請求的 deadline 作為排程的參考因素，因此 C_4 的請求不會超過 deadline，同時也滿足了其他的請求。

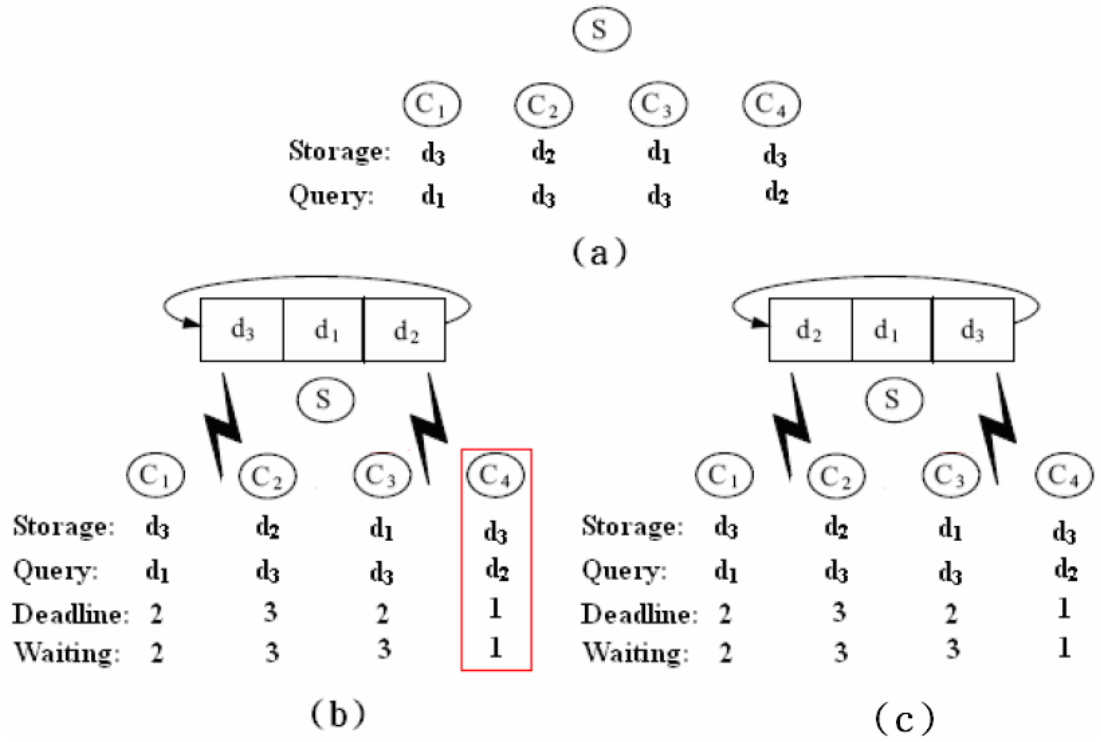

圖 3-4：Deadline 的問題描述圖

另外在真實的廣播環境中，頻寬的大小通常是有限的，因此在盡可能滿足最多請求的相關議題上[5] [6] [11]，主要是以資料項的冷熱門程度作為排程的考慮因素，請求頻率最高的資料項也就是最熱門的資料項，能為大部分的請求提供服務，因此應該最優先配置在廣播排程當中。

雖然在有限的頻寬配置了較為熱門的資料項以滿足大部分的使用者，但是在頻寬的大小固定不變的情況下，所能滿足的請求數量仍然有限，如圖 3-5（a）為一般的資料廣播排程，因此有些相關研究[3] [10]將網路編碼的技術應用在有限的頻寬上，使得有限的頻寬相較於以往能配置更多的資料項，進而服務更多的請求，如圖 3-5。傳統的網路編碼是將

多個資料項以線性組合的方式編碼成一個編碼資料項，編碼後的資料項與原本單一個資料項的大小相同，如圖 3-5（b）是傳統的網路編碼，在圖中可以看到傳統網路編碼是將三個資料項配對加上係數並組合成一個編碼資料項，但是 Client 端在解碼時，必須先存取兩個編碼資料項並且解聯立方程式才能解碼獲得所需要的資料項，其總和存取時間為 10。圖 3-5（c）是採用資料子集的編碼方式，將兩個資料項編碼成一個編碼，代表著 Client 端只需要編碼資料項中的其中一個資料項就可以進行解碼獲得所需要的資料項，其總和存取時間為 9。圖 3-5（d）是 OE 演算法的編碼方式，將沒有作用的編碼資料項排除，來達到改善排程效率的目的，使得下個排程可以較快進行廣播，並節省頻寬的消耗，其總和存取時間為 7。在此我們發現 OE 並沒有考量到請求具有時效性的問題。

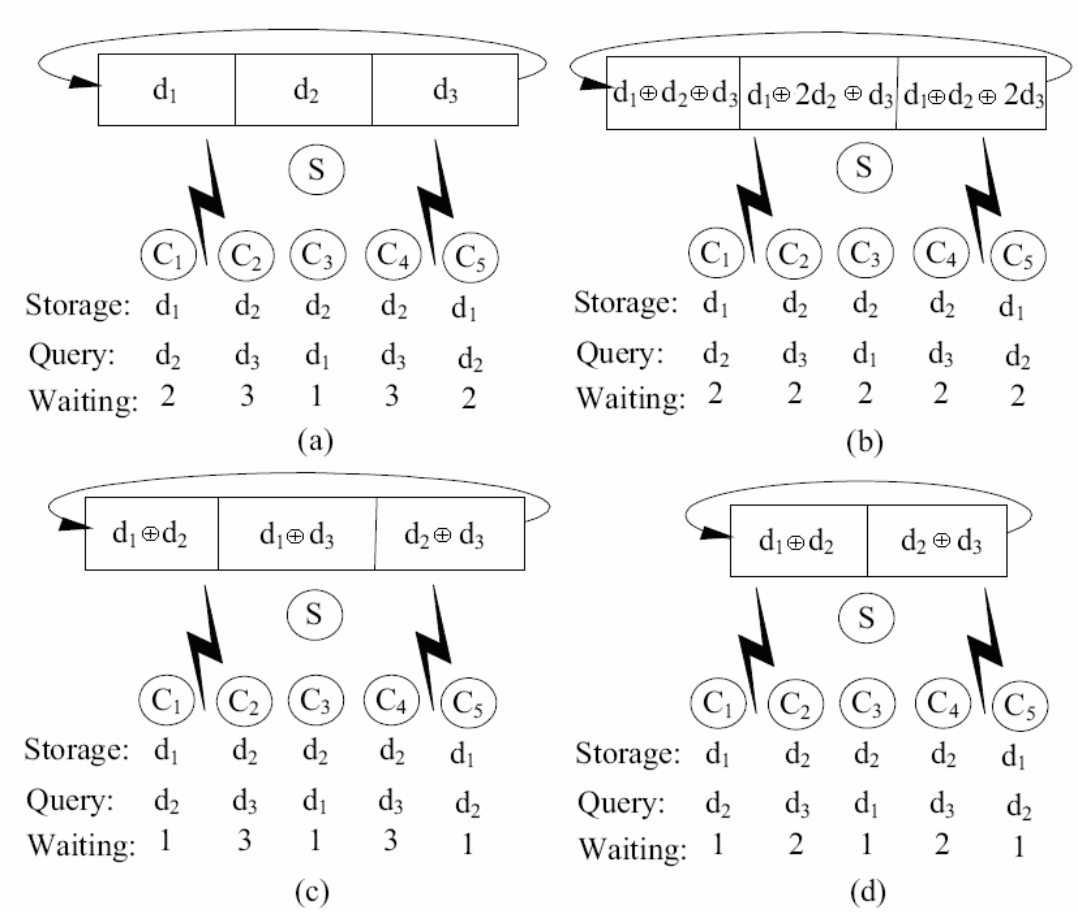

圖 3-5：編碼的問題描述圖

第四章、我們的演算法

我們提出一個有效率的排程演算法為 On-demand with Deadline Encoding(ODE)，本研究是以 OE 演算法為基礎架構，將請求的時效性作為排程的參考因素，可以提高請求的成功比率，不同於 OE 演算法將請求的等待時間作為排程的參考因素。下列將詳細介紹 ODE 演算法的運作。

第一節　主要概念

如先前所述，本研究所提出的演算法的主要目的是提高請求成功的比率、降低平均存取時間與滿足最多的使用者。為了解決上一章所提到的問題，因此將以等待時間為排程參考因素的 OE 演算法修改為以 Deadline 為參考因素的排程演算法 ODE。

在 OE 方法的基礎上，已經將資料項的請求頻率也納入考量並且在有限的頻寬下，利用編碼技術使得一個 time slot 可以一次服務多個請求並且降低頻寬的消耗，因此可以降低平均存取時間與滿足最多的請求數，而本研究將請求的 Deadline 納入考量是希望能從提高請求成功的比率、降低平均存取時間與滿足最多的使用者，這三個主要訴求當中取得一個平衡點並且更貼近於真實的廣環境。

第二節　運作程序

在此介紹整個系統的運作程序，當系統開始運作時，本研究假定 Client 端本身已經事先 Cache 了一些資料項，因此在 Client 端發出請求時，Client 端也會同時將本身所 Cache 的資訊一併傳送給 Server 端。Server 在接收到 Client 端所傳送過來的訊息後，會將 Client 端的請求與 Cache 的資訊以表格的方式存取在佇列中等待服務。接下來根據演算法產生排

程並廣播出去，而已經被服務完成的請求將會從等待服務的佇列中移除。

當 Client 端在傾聽廣播排程時，Client 端會從封包的標頭所包含的資訊來判斷是否有需要的資料項以及 Client 端本身是否有相對應的資料項可以進行解碼，如果沒有則忽視該封包，如果有就將封包接收下來。Client 端會利用封包的標頭所包含的解碼資訊，來解碼封包裡面的編碼資料項，解碼後的資訊會經由 CPU 轉換並透過螢幕來呈現出我們所看的懂得資料，同時解碼後所獲得的資料項會被 Cache 起來，以便 Client 端下次有相同需求時，可以直接從 Cache 中讀取。如果在資料項被 Cache 的過程當中，Cache Buffer 已經滿了，則將較沒有使用到的資料項剔除，再放入新的資料項，在這裡本篇是採用 OS 機制的中的 Least Recently Used(LRU)演算法[17]作為替換 Cache 資料項的策略。

接下來介紹我們所提出來的 ODE 演算法的運作程序，其運作程序大致上可以分成三個部分，分別為選擇所要服務的請求、資料項的編碼與排程、廣播與更新。第一部分選擇所要服務的請求，Server 使用公式（5）來計算在 Server 等待佇列中還未被服務的請求的權重植。公式（5）所產生的權重值是根據請求的頻率、Deadline，這兩個因素所產生。

第二部分為資料項的編碼與排程，選擇好優先服務的請求後，會將該請求的資料項與其本身所 Cache 的資料項，進行兩兩配對成編碼資料項，而這些編碼資料項納入排程的順序是依據公式（6）所求得的 $DE(e_x)$ 值，來代表編碼的權重，權重值越大的越優先納入廣播排程。如果編碼資料項未能將廣播週期排滿，則繼續進行第一部分與第二部分的運作，直到所有廣播排程都被塞滿為止。

第三部分是廣播與更新，將排程廣播出去後，Server 會從佇列裡移除已經服務的請求並更新這段時間到達 Server 端的請求與 Cache 的資訊。

第三節　ODE 演算法

本文所提出的 ODE 演算法的程式虛擬碼如下：

Input: The arrival users $u_1, u_2, \ldots, u_{n_u}$, the deadline $t_1, t_2, \ldots, t_{n_u}$ for each user after issuing a request, the sets of stored data items $S_1, S_2, \ldots, S_{n_u}$, the broadcast cycle length L, and the sets of requested data items $Q_1, Q_2, \ldots, Q_{n_u}$.

Output: The broadcast program P.

begin

1. **while** (there is any request arriving at the server)
2. Create an empty set P';
3. **while** $(|P'| \le L)$
4. **for** $(\text{each requested set } Q_i, 1 \le i \le n_u)$
5. Calculate $Store(S_i)$;
6. Calculate $Req(Q_i)$;
7. Calculate p_i for Q_i;
8. Serve the outstanding request Q_k first whose p_k is the largest one, where $1 \le k \le n_u$;
9. **for** $(\text{each } d_m \text{ in } Q_k, \text{ where } 1 \le m \le |Q_k|)$
10. **for** $(\text{each } d_n \text{ in } S_k, \text{ where } 1 \le n \le |S_k|)$
11. Calculate $e_x = d_m \oplus d_n$,
12. where $1 \le x \le |Q_k| \times |S_k|$;
13. Calculate $DE(e_x)$;
14. Find the encoding data with larger DE, and allocate the encoding data into the broadcast program P' such that Q_k is satisfied;
15. **if** $(|P'| \ge L)$
16. **return** P
17. **else**

end

圖 4-1：ODE 演算法的虛擬碼

第四節　以實例說明 ODE 演算法

從上述 ODE 演算法的程式虛擬碼，可以詳細的了解 ODE 演算法的演算過程，因此以下將透過實際例子來講解 ODE 演算法的運算過程，了解 ODE 演算法的優勢，如圖 4-2。

首先我們假設資料庫 D 中有六個資料項 $(d_1, d_2, \ldots, d_6)$，每個資料項大小均為 1Mbytes 並且廣播每個資料項均要耗費時間一秒；廣播頻道大小為兩個時間槽，每個時間槽大小均為 1Mbytes。在圖 4-2（a）中，每個使用者 u_i，所送出的請求包含客戶端請求資料項的集合 Q_i 和每個客戶端本身儲存資料項的集合 S_i 的資訊及客戶端發出請求的 Deadline，因此透過圖 4-2（a），我們可以清楚知道客戶端請求的詳細資料。

在圖 4-2（b）中，ODE 演算法會優先服務尚未服務的請求，所以會利用公式(3)和(4)計算 $Req(Q_i)$ 與 $Store(S_i)$ 值，接著再利用公式(5)計算 P_i 值，找出擁有 P_i 最大值的請求。在本圖例中擁有 P_i 最大值的為 Q_3，因此 ODE 演算法會根據 Q_3 本身請求資料項集合中的資料項和客戶端本身儲存資料項集合中的資料項來進行資料項的編碼配對，如圖 4-2（c）。

在圖 4-2（c）中，將使用者 u_3 請求的資料項和本身所儲存的資料項，作編碼配對的組合，利用公式(6)計算編碼資料項的 $DE(e_x)$ 值，當中 $d_3 + d_5$ 和 $d_1 + d_2$ 的急迫性最大，因此優先將編碼資料項 $d_3 + d_5$ 和 $d_1 + d_2$ 納入廣播長度為 2 的廣播排程當中，如圖 4-2（d）。假如排程仍有空位則根據編碼資料項的 $DE(e_x)$ 值的大小，將編碼資料項從大到小依序排入排程，直到排程沒有空位。此外如果頻寬相當的大，在請求 Q_3 所有的編碼資料項全部

配置在排程上後，仍有空位則由擁有次高P_i值的請求來作編碼並排程直到排程沒有空位為止，如有重複的編碼資料項就放棄，不再放入排程中。

Request	Storage	Request	Waiting time (time slot)	Deadline (time slots)
Q_1	$S_1=\{d_2,d_4\}$	$Q_1=\{d_1,d_5,d_6\}$	2	5
Q_2	$S_2=\{d_1,d_3,d_4\}$	$Q_2=\{d_2,d_5\}$	1	2
Q_3	$S_3=\{d_1,d_4,d_5\}$	$Q_3=\{d_2,d_3\}$	1	1
Q_4	$S_4=\{d_2,d_5\}$	$Q_4=\{d_1,d_3\}$	2	1

(a)客戶端發出請求的資料項、客戶端儲存的資料項、waiting time

Request	$Store(S_i)$	$Req(Q_i)$	p_i
Q_1	5/4	5/4	5/16
Q_2	3/2	1	3/4
Q_3	7/4	1	7/4
Q_4	1	1	1

(b)計算每個請求的p_i值

Encoding data e_j	$Dem(e_j)$	Total $\frac{1}{t_i}$	$DE(e_j)$
$d_1 \oplus d_2$	$\{u_1,u_2,u_3,u_4\}$	27/10	54/5
$d_1 \oplus d_3$	$\{u_3\}$	1	1
$d_2 \oplus d_4$	$\{u_2,u_3\}$	3/2	3
$d_3 \oplus d_4$	$\{u_3\}$	1	1
$d_2 \oplus d_5$	$\{u_1,u_3\}$	6/5	12/5
$d_3 \oplus d_5$	$\{u_2,u_3,u_4\}$	5/2	15/2

(c)取$DE(e_j)$最大值的優先納入排程

$d_1 \oplus d_2$	$d_3 \oplus d_5$

(d)第一次廣播排程

圖 4-2：ODE 演算法實例說明

在第一次的廣播排程廣播之後，Server 會更新 Client 端的請求與 Cache 裡的資料項，如圖 4-2（e），請求Q_1仍然尚未被滿足並且又有新的請求Q_5進來。在圖 4-2(f)中，ODE 利用公式(3)和(4)計算$Req(Q_i)$與$Store(S_i)$值，接著再透過公式(5)求得p_i並找出最大值p_i的請求Q_1。ODE 根據請求Q_1中的資料項和 Client 端本身所擁有的資料項來進行資料項的編碼配對並計算出每個編碼資料項 Deadline 的急迫程度$DE(e_x)$，如圖 4-2（g），急迫程度越大者代表 Deadline 即將到期，因此越優先納入排程當中，如圖 4-2（h）。在第二次的廣播結束之後，沒有新的請求產生加上所有 Client 端的請求都已經被滿足，所以 ODE 演算法停止運作。

圖 4-2 用來說明 ODE 演算法與圖 2-8 用來說明 OE 演算法的例子是相同的，因此我們可以去比較這兩個例子的排程結果。在存取時間上，我們所提出來的 ODE 演算法所耗費的存取時間為 4+2+2+2+2＝12；OE 的方法為 4+2+3+3+2＝14，因此我們得知我們所提出來的 ODE 方法與 OE 演算法在存取時間上是差不多的。在請求成功比率上，我們所提出來的 ODE 演算法讓所有的請求都在時效性之內被滿足；OE 的方法則是使請求Q_3與Q_4超過 deadline 而導致請求失敗，因此我們所提出來的 ODE 方法擁有較高的請求成功率。根據上述，我們所提出來的 ODE 演算法在存取時間上的表現不差，並且能提高請求成功的比率。

Request	Storage	Request	Waiting time (time slot)	Deadline (time slots)
Q_1	$S_1=\{d_1,d_2,d_4\}$	$Q_1=\{d_5,d_6\}$	4	3
Q_5	$S_5=\{d_1,d_3\}$	$Q_5=\{d_5,d_6\}$	8	4

(e)新的 request 產生與尚未被服務完成的 request

Request	$Store(S_i)$	$Req(Q_i)$	p_i
Q_1	2	2	4/3
Q_5	3/2	2	3/4

(f)計算每個請求的 p_i 值

Encoding data e_j	$Dem(e_j)$	Total $\frac{1}{t_i}$	$DE(e_j)$
$d_1 \oplus d_5$	$\{u_1,u_5\}$	7/12	7/6
$d_1 \oplus d_6$	$\{u_1,u_5\}$	7/12	7/6
$d_2 \oplus d_5$	$\{u_1\}$	1/3	1/3
$d_2 \oplus d_6$	$\{u_1\}$	1/3	1/3
$d_4 \oplus d_5$	$\{u_1\}$	1/3	1/3
$d_4 \oplus d_6$	$\{u_1\}$	1/3	1/3

(g)取 $DE(e_j)$ 最大值的優先納入排程

$d_1 \oplus d_5$	$d_1 \oplus d_6$

(h)第二次廣播排程，滿足了所有的請求

圖 4-2：ODE 演算法實例說明

第五章、模擬實驗

為了驗證本研究所提出來的 On-demand with Deadline Encoding(ODE) 演算法比 OE 演算法有較高的請求成功的比率並且在存取時間上有也不差的表現，我們透過模擬實驗來驗證。以下將分別說明模擬環境、模擬參數和實驗結果。

第一節　模擬環境

本研究的模擬環境是用 Visual Basic 6.0 作為撰寫模擬程式的語言，VB6.0 的優點在於語法簡單，並且可以快速的建立物件，減少開發的時間，因此我們採用 VB6.0 作為我們撰寫模擬程式的工具。

第二節　模擬參數

本研究的模擬環境的參數，使用表 5-1 的參數。

表 5-1：模擬實驗參數

參數	預設值	變動範圍
Number of Data Item	1000	--
Broadcast Cycle	100	50 ~ 200
Arrival Rate	5	5 ~ 20
Requested Length	10	5 ~ 20
Stored Length	10	5 ~ 20
Zipf	0.8	0 ~ 1

本研究的實驗將對照 ODE 與 OE 演算法在 Zipf、Mean Arrival Rate、Broadcast Cycle、Requested Length、Stored Length 這 5 種參數產生變化時的平均存取時間與請求成功的比率。這 5 種參數的實驗參數設定如表 5-2、表 5-3、表 5-4、表 5-5、表 5-6

表 5-2：比較 Zipf 分配的參數表

參數	設定值
Number of Data Item	1000
Broadcast Cycle	100
Arrival Rate	5
Requested Length	10
Stored Length	10
Zipf	0.2 0.4 0.6 0.8

表 5-3：比較 Mean Arrival Rate(request/time slot)分配的參數表

參數	設定值
Number of Data Item	1000
Broadcast Cycle	100
Arrival Rate	5 10 15 20

Requested Length	10
Stored Length	10
Zipf	0.8

表 5-4：比較 Broadcast Cycle 的參數表

參數	設定值
Number of Data Item	1000
Broadcast Cycle	50 100 150 200
Arrival Rate	5
Requested Length	10
Stored Length	10
Zipf	0.8

表 5-5：比較 Requested Length 的參數表

參數	設定值
Number of Data Item	1000
Broadcast Cycle	100
Arrival Rate	5

Requested Length	5 10 15 20
Stored Length	10
Zipf	0.8

表 5-6：比較 Stored Length 的參數表

參數	設定值
Number of Data Item	1000
Broadcast Cycle	100
Arrival Rate	5
Requested Length	10
Stored Length	5 10 15 20
Zipf	0.8

第三節　模擬實驗結果

模擬實驗比較 ODE 演算法與 OE 演算法的平均存取時間與請求成功的比率。我們分別測試當 Zipf 分配、Mean Arrival Rate (每個時間單位請求到達的比率)、Broadcast Cycle(廣播週期)、Requested Length(請求長度)、Stored Length(存取長度)這 5 種參數有變化時 ODE 演算法與 OE 演

算法的實驗結果。

從圖 5-1 來看改變 Zipf 分配對存取時間的影響，Zipf 越高代表資料項冷熱門程度的差異度越大；Zipf 越低代表資料項冷熱門程度的差異度越小，也就是說資料項的冷熱門之分越不明顯。圖中 Zipf 等於 0.8 時，平均存取時間是最低的；Zipf 等於 0.2 時，平均存取時間最高；因此我們得知當資料項的冷熱門程度差異越大時，平均存取的時間就越小。在改變 Zipf 參數對平均存取時間影響的實驗下，我們所提出的 ODE 演算法與 OE 演算法相比，約比 OE 演算法多了 5%的存取時間，因此 ODE 演算法在平均存取時間上也有不錯的表現。

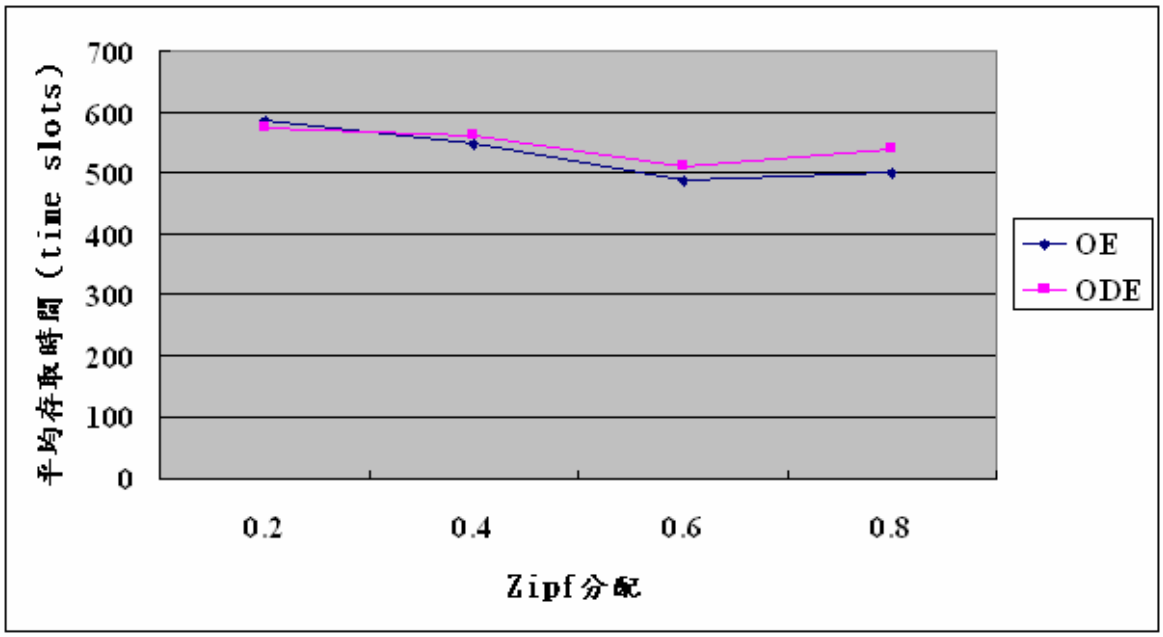

圖 5-1：平均存取時間_對照不同 Zipf 分配參數

圖 5-2 為 Zipf 分配請求成功的比率的影響，當 Zipf 越大相對的請求成功的比率也越大；從圖中我們所提出來的方法在成功比率上，比 OE 演算法多了 20%以上的成功比率。

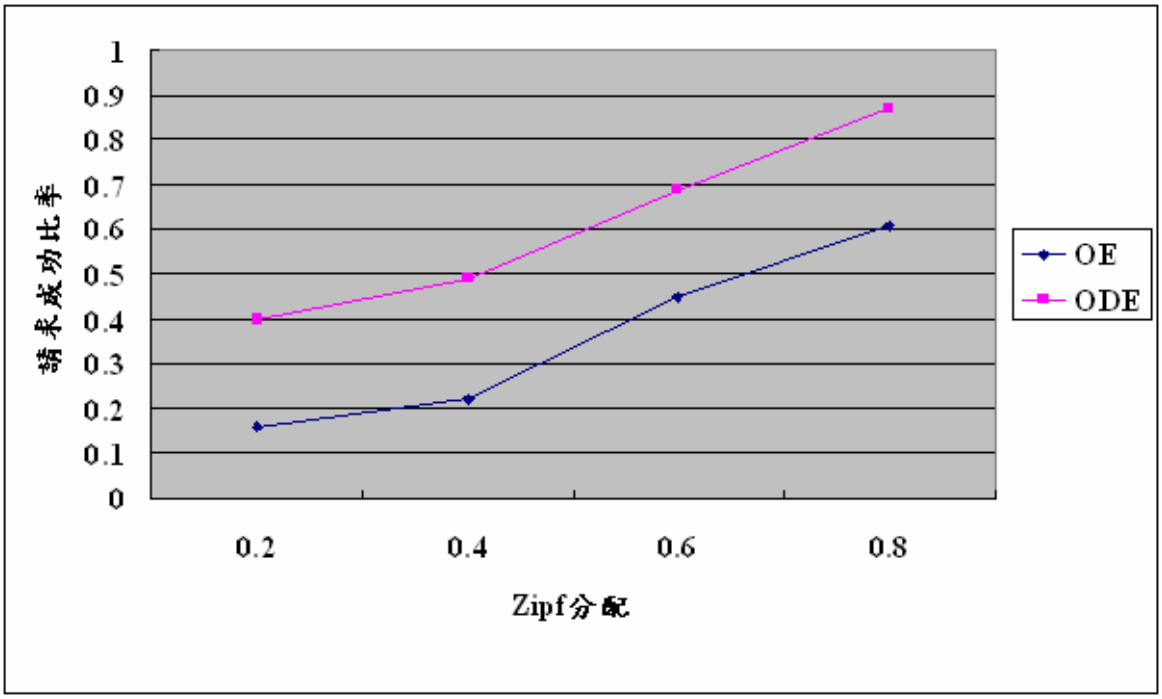

圖 5-2：請求成功比率_對照不同 Zipf 分配參數

Mean Arrival Rate(request/time slot)指的是每個時間單位請求到達 Server 端的個數。由圖 5-3 可知，隨著請求到達比率越高，平均時間也會跟增加，我們所提出的 ODE 方法與 OE 演算法的平均存取時間是幾乎相同的，兩者差異不大。

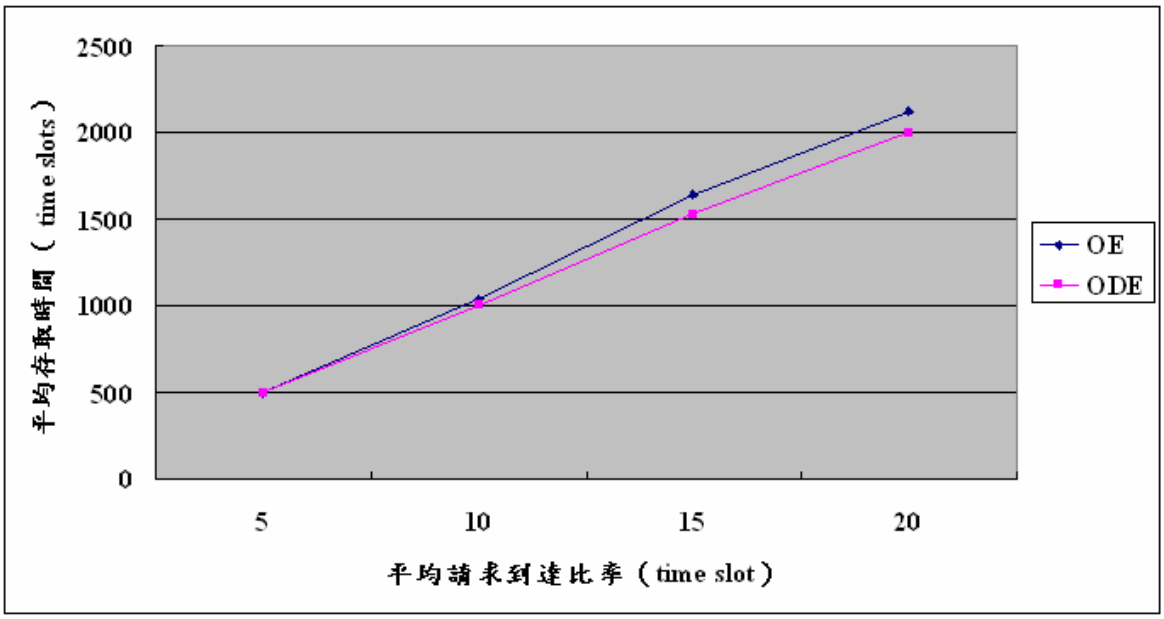

圖 5-3：平均存取時間_對照不同的請求到達比率

圖 5-4 中， ODE 與 OE 演算法並沒有隨著請求到達比率的增加而降低請求成功比率，是因為資料項冷熱門程度差異大時，大多數的請求都集中在熱門資料項，優先滿足熱門資料項的請求也就滿足了大多數的請求，因此 ODE 與 OE 演算法沒有隨著請求到達比率的增加而降低請求成功比率，ODE 在請求成功的比率比 OE 演算法多了 20%以上。

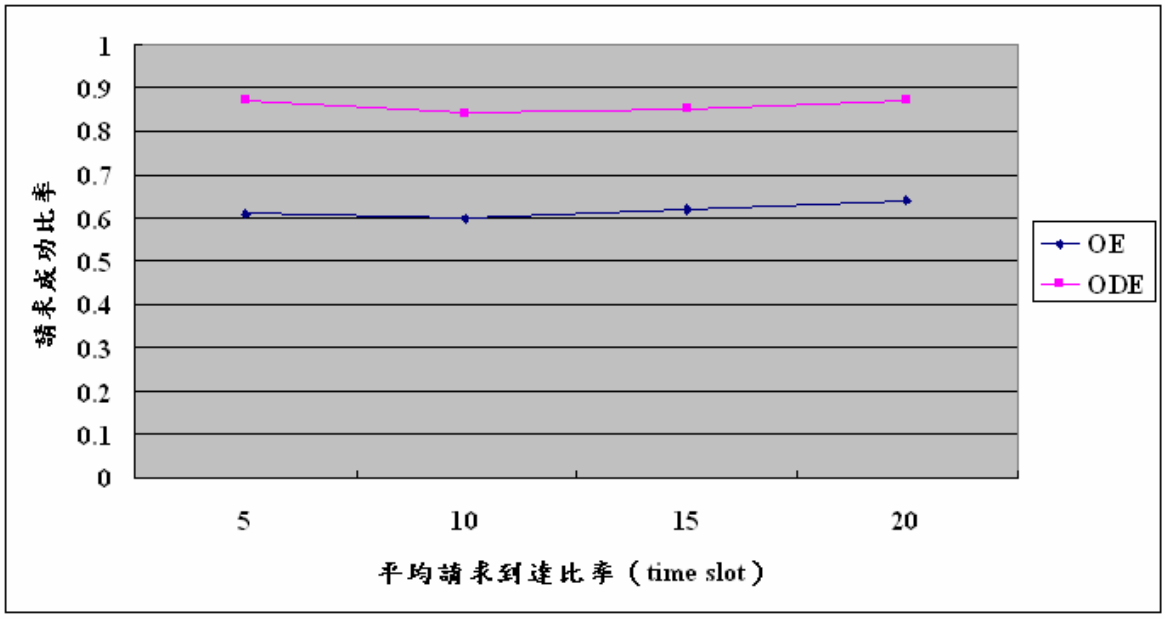

圖 5-4：請求成功比率_對照不同的請求到達比率

圖 5-5 為不同廣播週期長度的實驗，隨著週期長度的增加，所能滿足的請求數也跟著增加，因此存取時間也跟著增加，而我們所提出來的 ODE 演算法相較於 OE 演算法多了 5%的平均存取時間，因此 ODE 在存取時間上有不錯的表現。

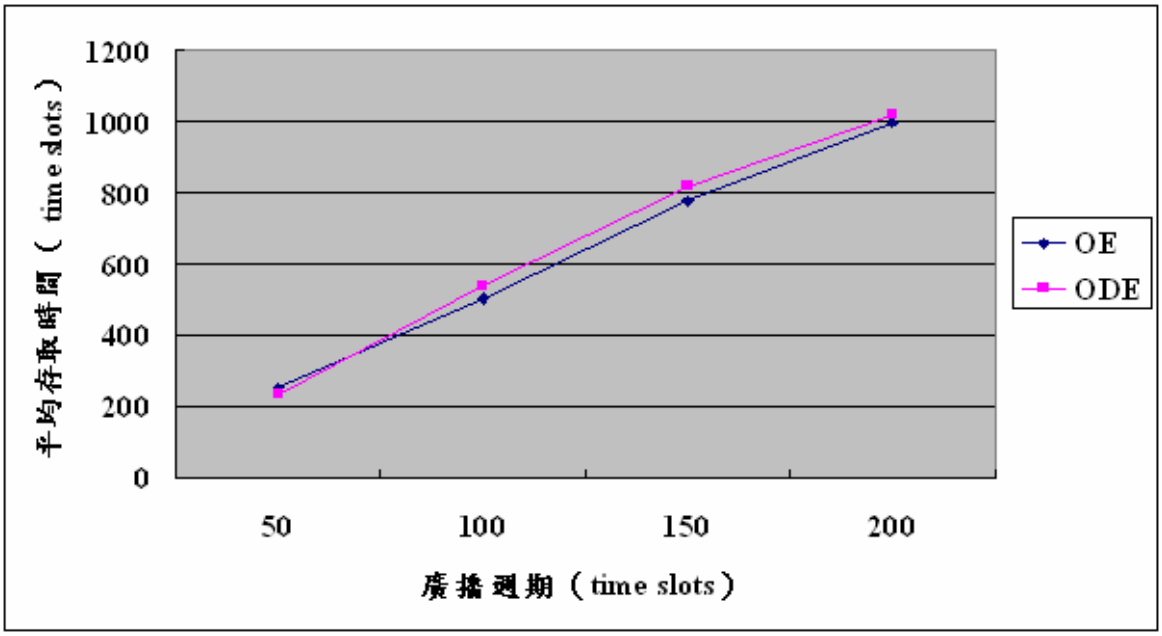

圖 5-5：平均存取時間_對照不同的廣播週期長度

圖 5-6 在不同的廣播週期長度下，實驗 ODE 與 OE 方法的請求成功比率。隨著週期長度的變長，容納的資料項也變多，因此能滿足更多的請求。從圖中可以看出，一旦頻寬夠大，我們提出的 ODE 和 OE 兩者的請求成功率最終會是相同的。

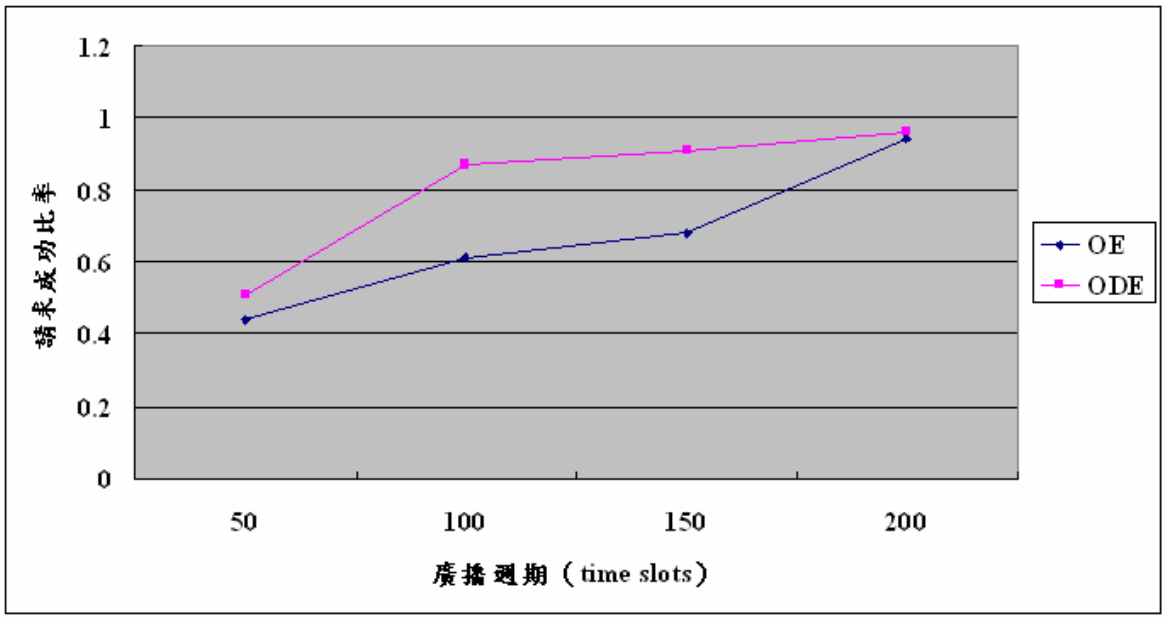

圖 5-6：請求成功比率_對照不同的廣播週期長度

從圖 5-7 來看，隨著請求長度的增加，也就是請求裡所包含資料項的增加，存取時間也會跟著增加。隨著請求長度的增加，我們所提出來的演算法 ODE 與 OE 相比，存取時間逐漸的增加，是因為 ODE 比 OE 滿足了更多的請求數。

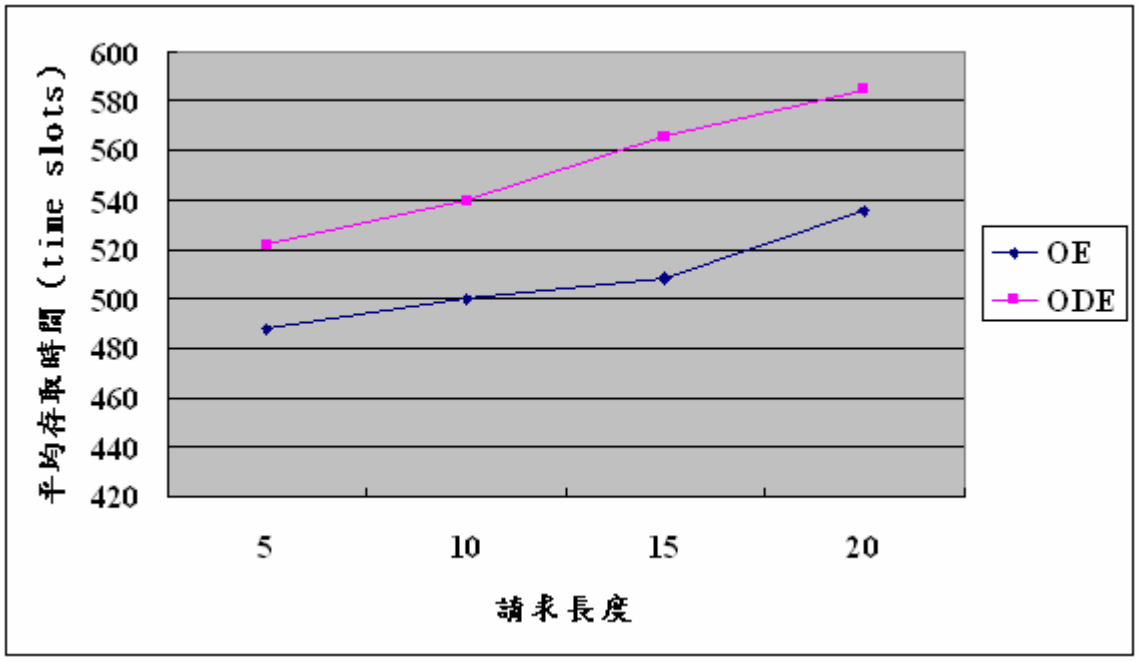

圖 5-7：平均存取時間_對照不同的請求長度

從圖 5-8 可以得知，隨著請求長度的增加，請求的成功比率會跟著降低，但是 ODE 始終保持著比 OE 更高的請求成功率。

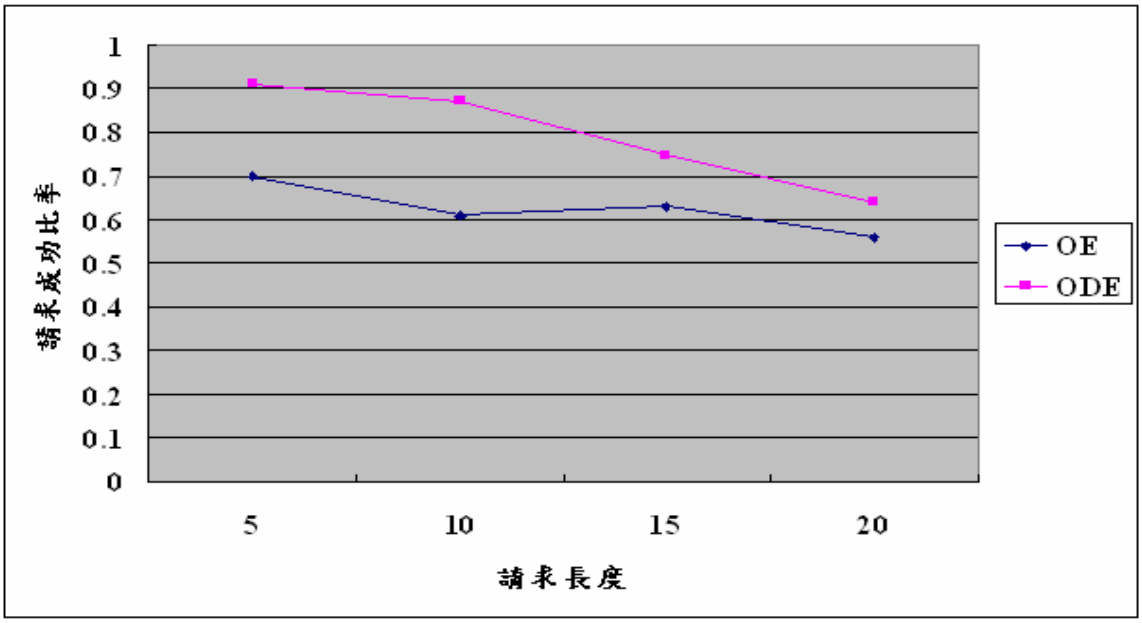

圖 5-8：請求成功比率_對照不同的請求長度

透過圖 5-9 我們可以得知，當 Client 端本身所 Cache 的資料項越多時，平均存取的時間會降低，因為假如當 Client 端有需要資料時，可以先從本身所儲存的資料項當中尋找，如果沒有再向 Server 發出請求；另外 Cache 的資料項越多，代表 Client 端可以解碼更多編碼資料項來獲得所需要的資料項。

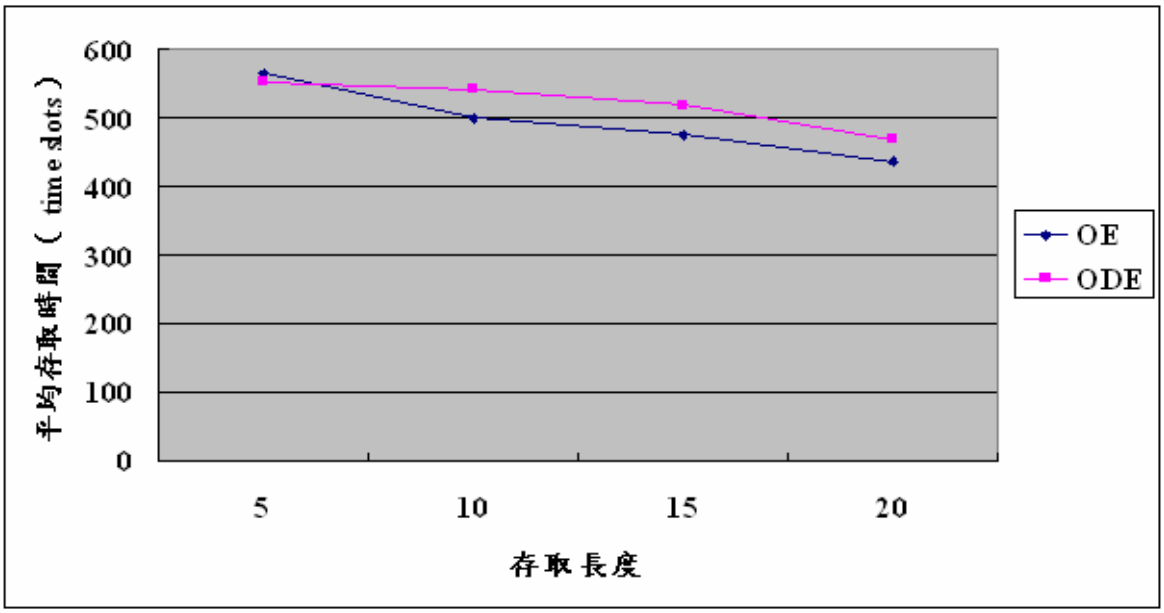

圖 5-9：平均存取時間_對照不同的存取長度

如上所述，Client 端 Cache 的資料項越多，代表 Client 端可以解碼更多編碼資料項來獲得所需要的資料項，因此可以提高請求被滿足的成功比率，也就是說存取長度越長，請求的成功比率也越高，如圖 5-10，而我們所提出的 ODE 演算法的請求成功比率都比 OE 演算法來的高，尤其是在存取長度越短的環境下。

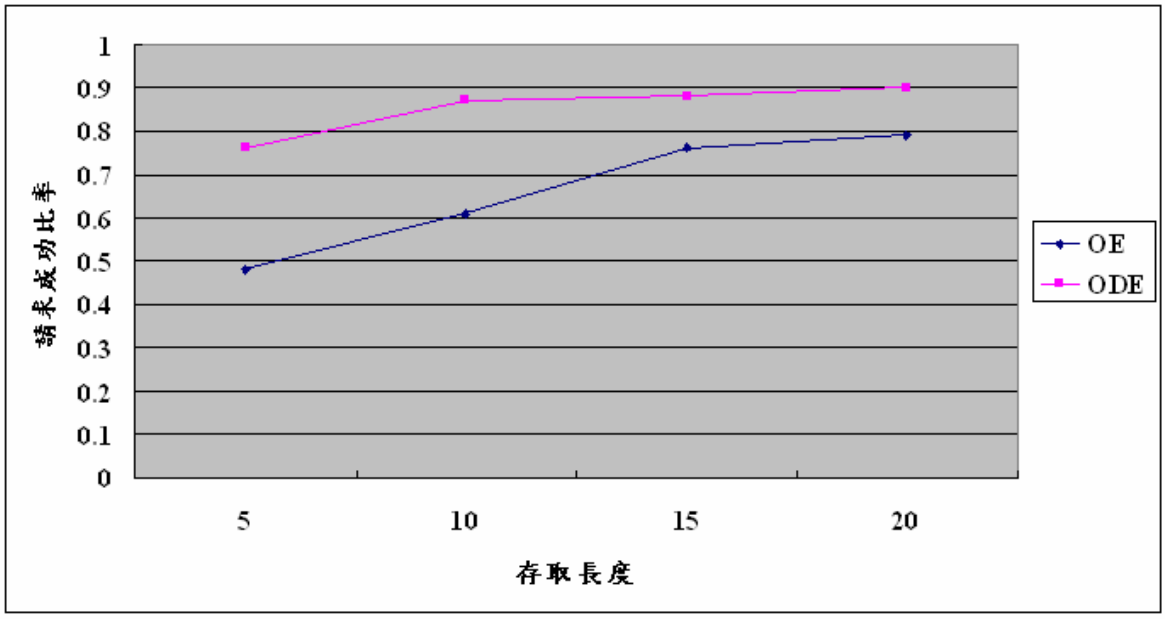

圖 5-10：請求成功比率_對照不同的存取長度

第六章、結論

在緒論的部份我們介紹了無線網路的背景與架構；在文獻探討部分，我們介紹了在不同條件限制下的先前研究；問題描述部分則是說明了以往的演算法未考慮到的地方；最後本研究提出了 On-demand with deadline encoding（ODE）演算法去改善先前所提出的 OE 演算法。

我們所提出的 ODE 演算法與 OE 做比較，在平均存取時間上，我們所提出的 ODE 演算法相較於 OE 的方法多了 5%的平均存取時間，因此 ODE 演算法在存取時間上也有不錯的表現；在請求成功的比率上，我們所提出來的 ODE 方法比 OE 多了 20%的成功率，也就是說能滿足更多的請求。根據上述 ODE 可以比 OE 滿足更多的請求並且在平均存取時間上跟 OE 相比也能有不錯的表現。

參　考　文　獻

[1] G. Lee,M. Yeh, S. Lo, and A. Chen. “A strategy for efficient access of multiple data items in mobile environments.” In Proceedings of the 3rd International Conference on Mobile Data Management, 2002, pages 71–78.

[2] Shih-yang Lin. “A Dynamically Schedule Algorithm for Wireless Broadcast.” June 2005.

[3] Chung-Hua Chu, De-Nian Yang and Ming-Syan Chen. “Multi-data Delivery Based on Network Coding in On-demand Broadcast.” Mobile Data Management, 2008. MDM 08. 9th International Conference on 27-30 April 2008, pages 181–188.

[4] Weiwei Sun, Zhuoyao Zhang, Ping Yu and Yongrui Qin. “Skewed Wireless Broadcast Scheduling for Multi-Item Queries.” Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on21-25 Sept. 2007, pages 1865–1868.

[5] Sang Hyuk Kang, Sujeong Choi, Seong Jong Choi, Gwangsoon Lee, Jaeug Lew and Jun Lee. “Scheduling Data Broadcast Based on Multi-Frequency in Mobile Interactive Broadcasting.” Broadcasting, IEEE Transactions on Volume 53, Issue 1, Part 2, March 2007, pages 405–411.

[6] Sujeong Choi, Sang Hyuk Kang and Yoon Goo Nam. “A Hybrid Broadcast Scheduling for Mobile Data Broadcasting with Return Channels.” Communications, Computers and Signal Processing, 2007. PacRim 2007. IEEE Pacific Rim Conference on 22-24 Aug. 2007, pages 549–552.

[7] D. Aksoy and M. Franklin, “R×W: A Scheduling Approach for Large-Scale On-Demand Data Broadcast.” IEEE/ACM Transactions on Networking, 7(6) 1999, pages 846–860.

[8] Jun Chen, Kai Liu and Victor C.S.Lee. “Analysis of data scheduling algorithms in supporting real-time multi-item requests in on-demand broadcast environments.” Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on 23-29 May 2009, pages 1–8.

[9] Yon Dohn Chung and Myoung Ho Kim, “QEM: a scheduling method for wireless broadcast data.” Database Systems for Advanced Applications, 1999 Proceedings. 6th International Conference on 19-21 April 1999, pages 135–142.

[10] Chung-Hua Chu, De-Nian Yang and Ming-Syan Chen, “Using Network

Coding for Dependent Data Broadcasting in a Mobile Environment.” Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE 26-30 Nov. 2007, pages 5346–5350.

[11] Kang, S.H., Soojeong Choi, Nam, Y.G. and Lee, J., “Combination of Push and Pull Scheduling for Mobile Interactive Data Broadcasting.” Communications, 2008. ICC '08. IEEE International Conference on 19-23 May 2008, pages 2001–2005.

[12] Jiun-Long Huang and Wen-Chih Peng, “An effective broadcast program generation algorithm for dependent data.” Proceeding of the IEEE 2005 Emerging Information Technology Conference, August 2007.

[13] Ming Lei, Susan V. Vrbsky and Yang Xiao, “A Cost Model for Scheduling On-Demand Data Broadcast in Mixed-Type Request Environments.” Proceeding of the Global Telecommunications Conference, November 2007, pages 514–519.

[14] S.-Y. R. Li, R. W. Yeung and N. Cai, “Linear network coding,” IEEE Transactions on Information Theory, vol.49, 2003, pages 371–381.

[15] P. Xuan, S. Sen, O. Gonz´alez, J. Fernandez, and K. Ramamritham, “Broadcast on demand : efficient and timely dissemination of data in mobile environments.” Proceeding of the 3rd IEEE Real-Time Technology and Applications Symposium (RTAS '97), June 1997, pages 38–48.

[16] J. W. Wong. Broadcast delivery. Proceedings of the IEEE, VOL.76, NO.12, December 1988, pages 1566–1577.

[17] E.J. O’Neil, P.E. O’Neil, and G. Weikum. The lruk page replacement algorithm for database disk buffering. In Proceedings of the 1993 ACM Sigmod International Conference on Management of Data, pages 297–306.