適用於無線網路環境中之序號更新廣播快取失效策略

A Cache Invalidation Scheme with Ordered Updated Invalidation Reports for Wireless Mobile Environments

伍朝欽, Chao-Chin Wu

國立彰化師範大學資訊工程學系

Department of Computer Science and Information Engineering

National Changhua Univ. of Education, Changhua, Taiwan

ccwu@cc.ncue.edu.tw

洪秉鈞, Ping-Chun Hung

國立彰化師範大學電機工程學系

Department of Electrical Engineering, National Changhua University of Education, Changhua, Taiwan

pchung@ms81.url.com.tw

摘要

G. Cao 已經發表數個以基於 UIR 的新技術來解決傳統上基於 IR 方式的快取失效管理策略所引發的兩個問題：查詢延遲時間過長與較差的頻寬使用率。然而我們發現目於這些技術均是採用同步廣播失效報告的方式來改善最慢查詢的回應時間，因此在高更新頻率時仍無法有效的縮短查詢回應時間。針對此項缺點，本論文提出以同步及非同步混合廣播失效報告的方式設計出一種新的快取失效策略，稱為「序碼更新廣播」快取失效策略。此策略主要是以週期性方式廣播失效報告，但改以非週期性方式廣播 UIR：伺服器會在更新發生時立刻廣播 UIR。根據模擬結果，相較於 UIR 策略在高更新頻率時的查詢回應時間，序碼更新廣播快取失效策略約可縮短 30%之時間。

關鍵詞：行動通訊、快取失效策略、延展性、省電、無線環境。

Abstract

G. Cao has proposed several novel techniques to address the two problems incurred by the invalidation report (IR)-based cache invalidation scheme: long query latency and poor bandwidth utilization. In his schemes, the server also periodically broadcasts multiple updated invalidation reports (UIRs) during two adjacent IRs. However, although the periodical method assures of the maximum query response time, the query response time can't be reduced well at high update frequency. Therefore, we propose a new cache invalidation scheme, called the OUIR strategy, to combine the advantages of periodic and aperiodic approaches. In the OUIR strategy, the server transmits a UIR as soon as an update occurs but still broadcasts IRs periodically. Compared with the UIR scheme according to the simulation results, the OUIR strategy can shorten the query latency about 30% when the mean update arrival time is 0.5 second.

Keywords：Mobile communication, cache invalidation scheme, scalable, power conservation, wireless environment.

一、緒論

相較於有線網路環境，在無線行動環境中傳輸資料包括了兩項非常重要之特性：行動端經常斷線與頻寬有限，因此如何在有限的頻寬下使用有效的機制來達到省電及提高資料傳輸的效率便成為一個非常重要的課題[1]。快取（Caching）技術即是其中一個能有效解決此問題的方法，此技術係將經常查詢的資料存在客戶端本身的快取記憶體中，如此一來，在往後的查詢中若需相同的資料，行動客戶端即可在快取記憶體中獲取資料，而無需重覆向伺服器端要求傳送相同的資料[1]。

在使用快取技術時，我們必須在伺服器與客戶端兩者間維持其資料的一致性。在有線網路環境中，我們可以在有資料更新時，藉由伺服器通知客戶端哪些資料已經產生異動，然後將客戶端中之快取記憶體的資料做廢或更新，以達到資料的一致性。但由於在無線環境中，客戶端可能因各種因素而自願（如：省電起見）或非自願（如：收不到訊號）的經常斷線[1]，因此無法保證客戶端可以收到所有伺服器所廣播出來之更新資訊，如此一來就無法獲知客戶端中之資料內容是否為最新版本。為了解決這個問題，已經有許多相關的研究陸續被提出。以伺服器有無記錄行動客戶端暫存資料來分類，分別為有記錄狀態伺服器（Stateful server）[4, 6]與無記錄狀態伺服器（Stateless

server）[2, 3, 5, 7-10]兩類：

I. 有記錄狀態伺服器：伺服器端必須記錄每個行動客戶端擁有那些暫存資料。有記錄狀態伺服器在無線行動環境中的應用是較複雜的，例如，每當快取記憶體中的資料被取代，客戶端必須通知伺服器。另外，當行動客戶端從一個區域移動到另一個區域時，必須將本身暫存資料的資訊上全部傳給新的伺服器端知道。更重要的是，此方式的建置成本會隨區域中的客戶端總數成長而同步成長，因此並不具延展性(Scalability)。

II. 無記錄狀態伺服器：伺服器端不需要記錄任何有關行動客戶端所擁有的暫存資料，行動客戶端透過伺服器端所廣播的失效報告(IR: Invalidation Reports)來確認快取記憶體中資料的有效性[1]。無記錄狀態伺服器最大的優點是它的建置成本與行動客戶端的數量無關，無論一個伺服器下有多少個行動客戶端都只需廣播一份失效報告。

另外，依照伺服端廣播資料異動訊息給行動客戶端的時間來分類，可分為同步（ Synchronous ） [1-6, 8-10] 及非同步（Asynchronous）[7]兩種：

I. 同步：伺服器端週期性地廣播失效報告。伺服器端保持追蹤更新記錄，並且定期地將資訊廣出去。即當伺服器端有資料發生更新時，伺服器端並不馬上將此更新訊息通知行動客戶端，伺服器端會先將更新資訊記錄下來，每隔一固定時間才將更新的資訊加入失效報告（IR: invalidation report）中廣播出來。此種方式的優點是客戶端在收到失效報告後即可判斷其快取記憶體中的資料狀態，因此使用者等待查詢結果所需的最長時間是固定的。但其缺點是(1)客戶端必須收到失效告才能決定如何回答使用者的查詢，因此平均查詢時間為廣播週期時間的一半，(2)在收到失效報告後，若確定所查詢之資料已經作廢，客戶端必須重新向伺服器要求最新資料。如此可能會讓許多客戶端同時在失效報告後向伺服器端發出請求，形成爭搶上傳頻寬之不利情形。

II. 非同步：一旦資料庫有資料更新立即就將資訊廣播出來。此方法對連線著的行動客戶端有效，允許它們立即被通知更新。但若有行動客戶端離線一段時間後又重新連線，行動客戶端就不知那些快取中的暫存資料在斷線時已經更新了。此種方式的優點是客戶端可以在第一時間得知資料已經異動，因此客戶端可以快速地回覆使用者的查詢。但其缺點是客戶端必須有一套機制來保證在連線時能接收到所有來自伺服器的資料異動訊息，而且在離線後重新連線時，必須有一套機制來確認哪些快取記憶體中之資料仍為最新版本。我們必須注意到：在無線網路環境中，斷線的情形之經常發生的。

因無狀態伺服器與同步的方式具有較優異的特性，因此目前絕大多數的研究均是屬於結合此兩類方式為主[1-3, 5, 8-10]。也就是伺服器無須紀錄各個客戶端的快取記憶體之資料資訊，而且其週性地廣播 IR，以通知客戶端將已經過時的資料作廢。然而 G. Cao 有鑒於此主流策略卻有著較長的回應時間以及較無效率的頻寬使用率等兩項缺點，而提出在連續的兩個 IR 間，週期性地廣播多個額外的 Updated Invalidation Reports (UIRs)[2, 3]。每個 UIR 中容只加入最近廣播 IR 後所更新的資料，利用這般的設計，UIR 的資料量會比 IR 的資料量小的多，因此不但不會造成帶大的網路負擔，而且客戶端在收到 UIR 後也可以決定如何回復使用者的查詢。但是我們從其實驗結果發現，UIR 在高更新頻率下，仍存在著較長的回應時間，因此本論文提出一個能進一步縮短回應時間的失效策略。

在我們的策略中，伺服器仍舊以同步的方式廣播 IR，但是改以非同步的方式來廣播 UIR：也就是每當有更新發生即發出相對的 UIR。雖然這樣的策略可以改善高更新頻率狀況下的反應時間，但若更新頻率太低，則可能會導致等待的時間過長，因此我們加上一系統參數：約定時間。若約定時間到達，但仍無更新產生，則發出一空白 UIR，以通知客戶端到目前為止，資料均無更新，因此可以利用快取記憶體中之資料來回答使用者之查詢。另一方面，對於無線網路環境中，客戶端經常會無法收到伺服器所傳播的訊息這項特點，我們讓每個 UIR 都有自己的序號，然後藉由是否有跳號來判斷是否有 UIR 沒有收到，若是，則必須等到收到下一個 IR，才能回答使用者的查詢。詳細的策略內容與模擬結果在後面的論文中，將會有詳細的說明。

本論文後面的內容組織如下：第二節介紹相關之研究，包括 IR，UIR 以及非同步之廣播方式，第三節正式說明序號更新廣播失效策略，第四節則是以模擬的方式評估新策略的效能，最後，第五節則是本研究的結論。

二、相關研究

在本節中我們將依序介紹以下三種技術相關研究：IR-based[1]的快取技術[8]、UIR 快取技術[2-6, 9]與非週期性 IR 之失效方法[8]。

2.1 IR-based 的方式

在許多研究中[1-10]顯示 IR-based 在無線行動環境中於快取記憶體的管理上是一個吸引人的方法。此方法係利用伺服器端定期地將資料更新的資訊廣播出來，而不是行動客戶端直接向伺服器端提出請求來得知快取暫存資料的有效性。行動客戶端是處於聽者（Listen）的角色，等待來自伺服器端的 IR 資訊，使用它們來驗証本身快取暫存資料的有效性。

IR-based 的方式是以週期性的方式來廣播失效報告（IRs），而失效報告（IR）是用來記錄更新的資料項，並且只有在伺服器端可以更新資料項。每一個行動客戶端都必須維持快取記憶體的一致性，因此它們都要等待 IRs，並且使用 IRs 來使它們的快取記憶體資料失效。為了正確地回答查詢，行動客戶端在未收到下一個 IR 之前，無論此時快取記憶體中的資料是否有效，都不可用快取記憶體中的資料回答查詢。因為我們不知道從上一次 IR 到現在這段期間要查詢之資料是否已被更新。我們必須在收到下一個 IR 之後，加以驗証，若快取記憶體的資料項是有效的，立即就可以回應這個查詢，否則，就需要向伺服器端來請求這份已更新的資料。

在時間戳記（TS：Timestamp）的方法中[8]，伺服器端每隔 L 的時間廣播一次 IR，而且每一個 IR 中都包含有一個現在的時間戳記 T_i 及一長串的 (d_x,t_x) 資料，其中 $t_x > (T_i - w * L)$。這裡的 d_x 代表著資料項的編號，而 t_x 是此物件最後一次更新的時間。w 則代表 L 的倍數，意指在一個 IR 的內容中，包含了多久時間內更新的物件資料項。當一個行動客戶端離線的時間超過（w * L）的時間，就不能再使用原來快取記憶體中的資料，必須將之全部清空。

2.2 UIR-based 快取失效方式

為了減少查詢的延遲，Cao 提出一個新的方法將 IR 擴量 m 次，也就是讓 IR 每次重覆出現的時間在(1/m)th 個 IR 的廣播區間內 [2, 3]。與先前 IR 的方式比較起來， 客戶端最多只需要等待(1/m)th 個 IR 的廣播區間的時間就可回應使用者查詢。所以，當不考慮處理時間（Processing time），與 IR 比較起來，則查詢的延遲時間可縮短 m 倍。

倘若，IR 包含大量的歷史資訊，而在一個 IR 的廣播區間中，又完整地重覆 m 次，將花費大量的下載頻寬來廣播 IR 造成浪費。因此，為了節省頻寬，在方法上稍做改變，將在一個 IR 區間內插入 m-1 個更新週期的失效報告（Updated invalidation report），簡稱 UIR。而每個 UIR 內容只加入最近廣播 IR 後所更新的資料，利用這般的設計，UIR 的容量會比 IR 的容量小的多。如同客戶端下載 IR 來辨識快取的有效性一樣，UIR 也可以被使用來做辨識。

2.3 非週期性 IR 之失效方法

The Algorithm based on Aperiodic IR（AAI），為一非週期性 IR 廣播演算法[7]，。此方法依 stateless-based server 方式所設計的快取失效策略，以非週期性的方式將 IR 廣播，且支援處理唯讀的交易（Read-only transaction）。對於非週期性 IR 方式而言，有個很嚴重的問題是「不確定的等待時間」，也就是說，由於 IR 的廣播時間不確定，造成使用者提出的查詢無法保証在一確定時間內回覆，使等待回覆的時間上限無限延伸。相反地，週期性廣播 IR 的方式，保証其最久的等待時間為 IR 廣播的週期時間，而平均的回覆時間為 IR 廣播週期時間的一半。因此，為了解決「不確定的等待時間」這個問題，AAI 支術除了以非對稱的 IR 廣播之外還增加了週期性的 GM（Guide message）來實行，以避免等待下個失效報告時不確定的等待時間。

伺服器廣播失效報告 IR，其 IR 的內容包含過去一段固定時間內更新的資料及更新序碼（Sequence number）。每當伺服器完成一筆更新處理（Update transaction），伺服器即廣播一次 IR，而客戶端是必須在收到 IR 之後才能對交易處理做完整的檢查，因此，若在無法確定下個 IR 到達時間的情況下，將會有嚴重的問題發生。雖然快取過的資料在離線及重新連線時是允許被使用的，但在離線前已經使用快取記憶體所完成的交易，仍然必須等待下個 IR 的到來才可回覆，若此時資料更新的頻率過低時，則此交易的延遲時間就會相當久。為了解決此問題，作者以一個新的控制訊息 GM，伺服器以週期性的方式廣播 GM，而 GM 的內容只包含上一次 IR 的更新序碼。如此一來，在客戶端重新連線時快取記憶體有效性的確認，可藉由非週期性的 IR 及週期性 GM 來完成。

AAI 的 IR 內容包含最近更新的資料項外還以捎載的方式攜帶先前更新的資料項。然而，AAI 的 IR 內容與 IR-based 和 UIR-based 不同的是，IR-based 和 UIR-based 的 IR 內容是根據某一固定時間內更新的物件項才會加入到 IR 的失效報告中，而 AAI 則是包含固定更新次數的 IR 內容，若 IR 包括了三次 IR 更新

的內容，其 IR 內容除了這次更新的資料項外還加入了前兩次更新的資料項。如此一來，IR 所記載的更新資料項不再以固定的時間做為依據。因此，若 IR 更新的次數不那麼頻繁，則 IR 可記載更新資料項所更新的時間可能比 IR-based 或 UIR-based 所記載的時間要長，但若資料項更新的過於頻繁，則會使記載更新的資料項少於 IR-based 或 UIR-based 的IR 內容。

三、序號更新廣播之快取失效策略

3.1 序號更新廣播技術

在伺服器廣播傳送資料的技術上可分為同步及非同步傳輸，在先前所提的 IR-based 及 UIR-based 的失效方式即是以同步傳輸的技術，而 AAI 的快取失效方式則是以非同步的技術來實踐。若以同步傳輸技術而言，其優點為有較固定的查詢延遲時間，其平均的查詢延遲時間為傳輸失效報告週期時間的一半。但其缺點亦為有太固定的查詢延遲時間，而導致查詢延遲時間無法縮短。因此，我們以非週期性的方式廣播 UIR，希望藉此可縮短查詢延遲的時間。

在我們序號更新廣播的動作中，當客戶端收到伺服器所廣播的失效報告後，對於使用者所提出的查詢始可回應，若使用者所提出的查詢物件不在快取記憶體中或在快取記憶體中此物件已失效，則客戶端將此查詢物件 id 送出。然而，我們提出此立即更新廣播的方式與 UIR-based 的方法最大的不同在於 UIR 失效報告廣播的時間，UIR-based 是以週期性地廣播方式傳送，而我們的方式是以非週期性地方式廣播 UIR 失效報告，而且當有物件更新時就立即廣播，而且每次廣播 UIR 時都加上一個序碼，用以判識 UIR 的順序，而且在 IR 廣播之後，序碼重新歸零。故稱此快取失效策略為「序碼更新廣播之快取失效方法（OUIR：Ordered Updated Invalidation Report）」。以下我們詳細說明此快取失效策略。

我們仍維持以週期性的方式來廣播 IR，若資料庫中資料有任何資料變更，除非伺服器正在廣播其它的物件，否則在第一時間，伺服器會將此資料項 id 廣播，若伺服器正在廣播物件，則等待此物件廣播完畢後，立即廣播此更新的物件 id。但是為了避免在資料庫資料更新頻率低的情況下，造成查詢回應時間的延長（最長為 IR 廣播的週期時間），我們設定一個約定時間，用來約定下個 UIR 最慢必須被廣播的時間，若在時間內無任何更新資訊被廣播，則伺服器將 UIR 的序碼加一，但更新資料項 id 內容為 NULL。若在 IR 廣播後於約定時間內無任何更新產生，則在約定時間到時廣播一個 NULL 的訊息，用以通告客戶端，在客戶端接到此訊息後仍可做回應查詢的動作。

另外，由於無線網路傳輸經常會有無法接收訊息的情況產生，因此為了避免更新資訊遺漏的情況，造成客戶端之資料沒有適當的更新，我們必須在 UIR 廣播的內容上做些調整。原本 UIR 的內容僅紀錄更新資料項 id，現在我們在每個 UIR 附加上一個序碼，而且每增加一個新的更新即發送 UIR 並將搭配的序碼值加一。若在約期時間內無任何資料更新，則廣播一為 NULL 的UIR，當客戶端收到時仍可提出查詢回應使用者。因此，若在一個 IR 週期時間內，客戶端可藉由此一連續的序碼值判斷是否發生 UIR 漏接的情況，若收到的序碼值不連續即發生跳號的現象就判定為 UIR 漏接的狀況發生，此時客戶端就必須等待下次 IR 到達時，始可對使用者的查詢做回應。仍然可以從現在這個接收到的 UIR 資訊來回應使用者的查詢。若是漏接 IR，仍必須等待接收下個 IR 始可對使用者查詢的物件做回應。以下我們以一範例，如圖一所示，而且假設 IR 的廣播週期為 20 秒，約定時間為 5 秒。

(1)、客戶端快取記憶體的初始狀態，一開始擁有的物件包括<A,B,C>其對應的時間戳記為<10,18,30>。

(2)、此時使用者對物件<a,b,d>提出查詢，檢查快取中資料項發現有 a 及 b 物件，卻沒有 d 之物件。注意此時 a 與 b 不可立即回應給使用者。且為了節省電能 miss 物件仍要累積到下個 IR 或 UIR 收到時才可上傳。

(3)、此時，伺服器廣播 UIR，其內容為物件 b。

(4)、客戶端接收 UIR 後。由於客戶端在步驟②時對物件<a,b,d>提出查詢，檢查快取記憶體發現物件 b 已失效，且快取記憶體中沒有物件 d 的資料，於是客戶端向伺服器提出物件 b 和 d 的請求，而 a 在快取記憶體中為有效的資料項，所以直接在快取中取得資料回應給使用者。

(5)、當伺服器收到客戶端的查詢後，立即將物件 B 及 D 回傳給客戶端。

(6)、當伺服器傳來查詢的資料物件時，客戶端將收到的資料放入快取記憶體中，若快取記憶體中沒有失效的資料項，則替換掉最久未更新的資料，所以，此時 A 被替換掉。

(7)、此時有物件 e 被更新，於是伺服器發

出失效報告 UIR，不過其內容包含了上個 UIR 的內容。

(8)、客戶端收到 UIR 後檢查快取記憶體，發現沒有任何物件失效，於是快取記憶體維持原狀。

(9)、伺服器於 70 秒時廣播 IR，此時沒有新的更新資訊。

(10)、客戶端收到 IR 後檢查快取記憶體，並無失效的物件。

(11)、於 85 秒時伺服器發現沒有任何資料被更新且約定時間也到，於是廣播一個 NULL 的訊息給客戶端。

(12)、若客戶端的使用者在 80 秒到 85 秒間有提出查詢，則可利用此 NULL 訊息廣播來回應。當客戶端收到此 NULL 訊息其快取記憶體的內容不變。

(13)、此時伺服器更新物件 d，並且立刻將此訊息廣播。

(14)、客戶端收到 UIR 後將快取記憶體中資料項 d 設為無效資料項。

3.2 演算法

3.2.1 客戶端之演算法

d_x：the x^{th} data item in the database.
D：the set of data items.
L_{bcast}：an id list that the server will broadcast after the IR.
S_k：the sequence number of $UIR_{i,k}$.
T_{now}：the current time.
T_e ：the due time.

(A) At interval T_i, construct IR_i as follows:
$IR_i = \{<d_x,t_x> \mid (d_x \in D) \wedge (T_i - L \cdot w < t_x \le T_i)\}$;
Broadcast IR_i and L_{bcast};
for each $d_x \in L_{bacst}$ do
broadcast data item d_x;
$L_{bcast} = \varnothing$;
$S_k = 0$;

(B) When an update occurs or $(T_{now} - T_{i,k-1}) \ge T_e$, construct $UIR_{i,k}$ as follow:
$S_k = k$;
$UIR_{i,k} = \{(d_x, S_k) \mid (d_x \in D) \wedge (T_{i,k-1} < t_x \le T_{i,k})\}$;
Broadcast $UIR_{i,k}$ after construct it.

(C) Receives a request ($L_{request}$) from client C_j
for each d_x do
$L_{bacst} = L_{bacst} \cup d_x$;

3.2.2 客戶端之演算法

$Q_i = \{d_x \mid d_x$ has been queried before $T_i\}$
t_{xc}：the timestamp of cached data item d_x
T_l：the timestamp of the last received IR.
$L_{request}$：an id list of the data items that a client requested from the server.
S_r：the sequence number of the last UIR received.

(A) When the client C_j receives IR_i and L_{bcast}
if $T_l < (T_i - L \cdot w)$ then
then drop the entire cache or go uplink to verify the cache (or use other techniques to deal with long connection);
for each data item $< d_x, t_{xc} >$ in the cache do
if $(< d_x, t_x> \in IR_i) \wedge (t_{jc} < t_{xc}))$
then invalidate d_x;
for each $d_x \in L_{bcast}$ do
if d_x is an invalid cache item
then receive d_x in cache;
$T_l = T_i$;
$S_r = 0$;
if $(Q \neq \varnothing)$ **then** query (Q_i);

(B) Receives a $UIR_{i,k}$
if (IR_i is missed $\vee$ $(S_k \neq S_r + 1)$)
then break;
$S_r = S_k$;
for each data item $< d_x, t_{xc}>$ in the cache
if $d_x \in UIR_{i,k}$
then invalidate d_x ;
if $(Q_{i,k} \neq \varnothing)$ **then** query($Q_{i,k}$);

(C) Procedure query（Q）
$L_{request} = \varnothing$
for each $d_x \in Q$ do
if d_x is a valid entry in the cache
then use the cache's value to answer the query;
else $L_{requer} = L_{request} \cup d_x$;
if $(L_{request} \neq \varnothing)$
then send request $L_{request}$ to server;
Use the received data item to answer the query;
Add the received data item to local cache.

3.3 討論

「立即更新廣播」的主要用意在於降低查詢延遲的時間，因為利用較短的失效報告廣播使查詢回應的時間縮短，但由於伺服器不定時地廣播 UIR 失效報告，導致客戶端無法經常保持在 sleep 的模式下，必須經常保持在 work 的工作模式，而反觀 UIR-based 之失效策略，

由於失效報告廣播的時間固定，只要在固定的時間 wake 於工作模式下動作外，其它的時間皆可保持在 sleep 的模式下。因此，以我們「序碼更新廣播」的失效方式可能會比 UIR-based 的失效方式來得耗電。

然而，電能的消耗主要有以下幾個來源：(1)行動設備端使用者介面的操作，如顯示器所消耗的能量或按鍵時所造成電流的導通 (2)上傳資料(3)下載資料(4)資料儲存及系統相關消耗。以一般行動客戶端（如：手機、PDA）而言，無論是否要傳遞資料，只要使用者開始操作客戶端設備，此設備立刻由 sleep 模式進入工作模式，此時顯示器秀出操作畫面，系統被喚醒。而且使用者在操作設備時通常不會只用個幾秒鐘，可能比起傳輸資料的時間要長，因此，才許在使用者查詢下筆資料前就已經收到上次查詢的資料，所以在兩次查詢間不易進入 sleep 模式。另一方面，若使用者持續一段時間沒有操作客戶端設備，則系統自動再進入睡眠狀態。在「序碼更新廣播」方式下，只有在工作模式下，行動客戶端才會接收非同步的 UIR 訊息。意指，在行動設備連續操作的情況下「序碼更新廣播」與 UIR-based 的耗電量差距不大。

四、模擬環境與模擬結果

4.1 模擬環境之說明

我們根據 Cao[2]所設定的模擬方式，設計了一個模擬器，用來評估我們設計的快取失效策略之效能。此模型為一個伺服器對多個行動客戶端的架構。這模型包括數個模組，分別有：行動客戶端、伺服器端、上傳通道及下載通道。行動客戶端須經由上傳通道將查詢的請求傳輸給伺服器，而伺服器則透過下載通道將查詢的資料由傳給行動客戶端。行動客戶端會藉由接收 IR 或 UIR 來確認暫存資料是否有效，這才是透過上傳及下載通道來完成查詢的工作。我們所使用廣播的方法是參照 IEEE 802.11 的方式，在廣播下次 IR 或 UIR 的失效報告時，必須等到目前正在廣播的 packet 結束後方可廣播，然而這並不影響下次廣播 IR 或 UIR 的時間。我們在模擬開始之初，會先讓這模擬器運行一段時間，直到系統狀態穩定之後才採集相關的記錄，如：快取的命中率、猜到熱門資料的比例、查詢延遲時間及上傳資料傳送量。為了簡化模擬條件，我們將資料庫的大小、資料項大小、資料項 id 的大小、時間戳記的大小及上/下傳通道的頻寬等部份的參數值固定，而變動平均更新時間、平均查詢時間及門檻值等，其參數值可參考表一。

4.2 模擬結果

在此我們將 OUIR 之約定時間設定為 4 秒，UIR 複製 IR 週期時的倍數 m 設為 5，而 UIR 的廣播週期時間才是 4 秒。圖二為更新頻率對 OUIR 之快取命中率的模擬結果，圖三為更新頻率對 OUIR 之查詢延遲時間之模擬結果，圖四為更新頻率對 OUIR 之上傳量之模擬結果。從圖二快取命中率的結果中，可以觀察出 UIR 及 OUIR 無論更新頻率如何變動，皆有相同的快取命中率結果。這是由於我們 OUIR 的快取失效策略與 UIR 的差異，只有在 UIR 失效報告廣播時點上的差異，所以有相同的結果並不感到意外。但從圖三來看，尤其更新頻率在 2 秒以前，可發現我們 OUIR 的查詢延遲時間比 UIR 的查詢延遲時間縮短了 4 至 2 秒，這才就是我們設計 OUIR 最主要的目的，希望藉由高的更新頻率，造成快速回應使用者的查詢。而在低更新頻率時，OUIR 查詢延遲時間才比 UIR 好，平均都比 UIR 快 0.5 秒以上。而從圖四可發現 OUIR 與 UIR 才有相同的上傳量，這可以從快取命中率的結果看出端倪，由於上傳的資料只有查詢 miss 的資料項 id，因此有相同的快取命中率，才表示有相同的快取 miss 量，所以才造成有同樣的上傳量。由上傳量的結果可觀察出最重要的結果是客戶端的耗電量，由於客戶端在上傳資料時，要有足夠的功率才能讓基地台接收到訊號，故行動端需要消耗大量的電能將訊號發送出去，所以上傳量是行動設備電量消耗的標的。所以從圖四看來，OUIR 與 UIR 在主要的電才消耗上有相同的耗電量。

五、結論

本論文主要的設計目標是如何在無線行動環境中，設計一新的快取記憶體失效策略以改善 UIR 技術中高資料更新頻率狀況下查詢回應時間過長之問題。我們設計的方法是以週期性的方式廣播 IR，藉以保證離線的時間若在（w * L）內，客戶端的快取記憶體資料仍可以判斷是否與伺服器端的資料內容一致，而不需將全部的暫存資料全部作廢。另外，我們將 UIR 改以非週期性的方式廣播，每當有資料更新發生時，就立刻發出一 UIR，藉以希望在高更新頻率時，能獲得較佳之回應時間。但非同步廣播 UIR 的缺點是當更新頻率低時，可能會導致非常長的反應時間。針對此問題，我們設計了一個約定時間，若在時間內沒有任何的資料更新產生，則伺服器會傳送一個 NULL 的 UIR 訊息給客戶端，以通知其快取記憶體內的資料全部與伺服器一致，因此可以回答查詢，才就是此約定時間可以保證回應的最

長時間。另一方面，針對無線環境中經常性離線而無法收到更新資訊的問題，我們在每個 UIR 中加入一序號。客戶端可以藉由檢查是否有跳號來判斷是否有漏接任何的UIR。若無漏接，則可以決定如何回答查詢，否則必須等到收到下一個 IR 才能夠決定如何回答查詢。根據模擬結果，我們所提出的失效策略與 UIR 技術相比，有相同之快取記憶體命中機率以及上傳數量，但卻有著較短之查詢回應時間，尤其在高更新頻率下，可以縮短約 30%的等待時間。

參考文獻

[1] D. Barbara and T. Imielinski, “Sleepers and Workaholics: Caching in Mobile Distributed Environments,“ Proc. 1994 ACM-SIGMOD Int’l Conf. Management of Data, pp. 1-12, June 1994.

[2] G. Cao, "A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments," ACM MOBICOM'00, pp. 200-209, Aug. 2000.

[3] G. Cao, "A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments," IEEE Transactions on Knowledge and Data Engineering, Vol. 15, No. 5, September/October, 2003.

[4] G. Cao, "On Improving the Performance of Cache Invalidation in Mobile Environments," ACM/Kluwer Mobile Networks and Application (MONET), Vol. 7, no. 4, pp. 291-303, August 2002.

[5] G. Cao, "Proactive Power-Aware Cache Management for Mobile Computing Systems," IEEE Transactions on Computers, vol. 51, no. 6, pp. 608-621, June, 2002.

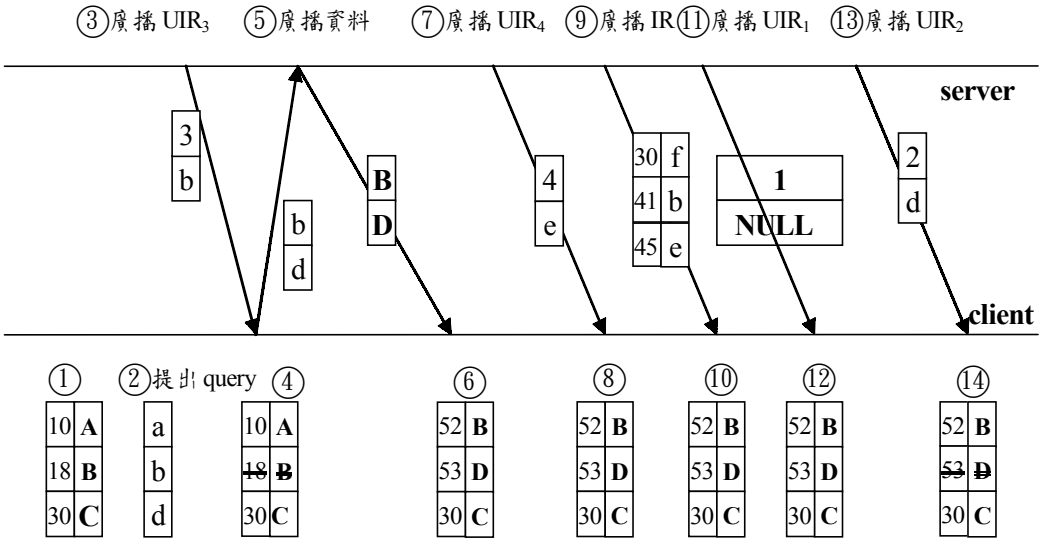
[6] G. Cao and C. Das, "On the Effectiveness of a Counter-Based Cache Invalidation Scheme and its Resiliency to Failures in Mobile Environments," The 20th IEEE Symposium on Reliable Distributed Systems (SRDS), pp. 247-256, Oct. 2001.

[7] SungHun Nam et. al., “An Efficient Cache Invalidation Scheme for Mobile Wireless Environments,” Intl. Conf. on Parallel and Distributed Systems, pp. 289-296, 2001.

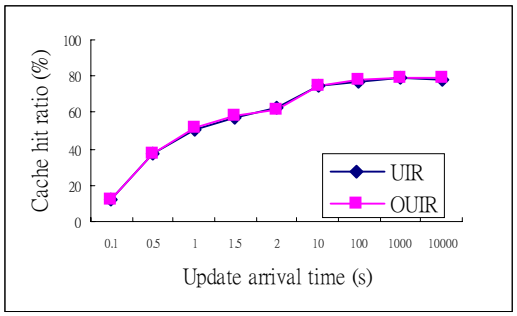
[8] Kina-Lee Tan, Jun Cai and Beng Chin Ooi, “An Evaluation of Cache Invalidation Strategies in Wireless Environments,” IEEE Transactions on Parallel and Distributed Systems, pp. 789-807, Aug. 2001.

[9] L. Yin, G. Cao, C. Das, and A. Ashraf, "Power-Aware Prefetch in Mobile Environments," IEEE International Conference on Distributed Computing Systems (ICDCS), pp. 571-578, July, 2002.

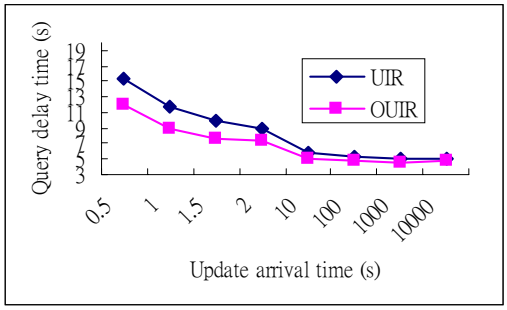
[10] J. Yuen, E. Chan, K. Lam, and H. Leung, “Cache Invalidation Scheme for Mobile Computing Systems with Real-Time Data,” ACM SIGMOD Record, Dec. 2000.



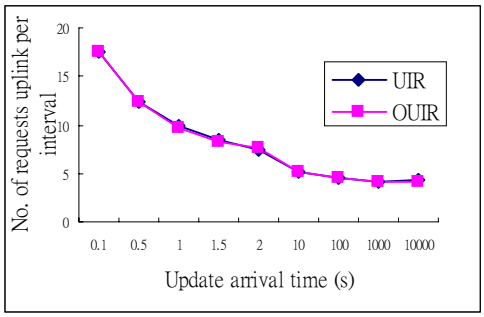
圖一：序碼更新廣播之快取失效方法之範例動作流程圖



圖二：OUIR 之快取命中率



圖三：OUIR 之查詢延遲時間



圖四：OUIR 之上傳量

表一：系統參數表

參數名稱	定義	預設值
DB_{size}	server 端資料庫容量	1000 obj
C_{num}	client 端的數量	100
C_{size}	cache 的容量大小	100 obj
t_q	平均 query 的間隔時間	100 sec
T_{q_dis}	選擇 t_q 分佈函數的參數	指數分佈
t_u	平均資料庫更新的間隔時間	100 sec
T_{u_dis}	選擇 t_u 分佈函數的參數	指數分佈
B_u	有百分之多少的物件是屬於熱門更新的集合 100%-B_u 為 cold update set 的大小	5
A_u	熱門更新的機率占百分之幾	33
B_q	有百分之多少的物件是屬於熱門查詢的集合 100%-B_q 為 cold query set 的大小	5
A_q	熱門查詢的機率占百分之幾	80
C_{up}	上傳頻寬的大小	2 kbps
C_{down}	下載頻寬的大小	10kbps
L	廣播 IR 的週期時間	20 sec
m	UIR 擺疊 IR 週期時的倍數	4
W	IR 記載更新歷史的大小	10
O	物件容量的大小	1K bytes
O_{id}	物件 id 所占容量的大小	32 bits
TS	時間戳記所占容量的大小	64 bits