Broadcast-IR:一個在資料廣播環境中的快取資料驗證方法

呂永和
國立臺灣科技大學資訊管理研究所
台北市基隆路四段四十三號
E-mail: yhl@cs.ntust.edu.tw

李建宗
國立臺灣科技大學資訊管理研究所
台北市基隆路四段四十三號
E-mail: M9309113@mail.ntust.edu.tw

摘要

在行動計算環境中,伺服端必須滿足大量客戶端的資訊需求，而使用資料廣播的資訊傳送方式，被視為是滿足大量客戶端資訊需求的好方法。在資料廣播的環境下，節省客戶端的能源消耗和降低客戶端取得資料所需的等待時間，是兩個相當重要的議題。客戶端快取（暫存）之前使用過的資料，是一個可以同時達到上述兩個目的的方法。但資料快取必須同時考慮資料驗證的問題：當伺服端改變其所儲存的資料項時，必須有一個方法通知客戶端哪些快取的資料項已經過時。本研究提出一個在資料廣播排程中加入驗證報告的方法，此方法可以降低取得驗證報告所需的時間，並能有效降低客戶端取得所需資料項的等待時間。在本方法中，我們修改了廣播封包的表頭，使得客戶端可以得知驗證報告廣播的時間；並根據資料廣播的特性，使用編號來取代傳統驗證報告中的時間戳記，降低驗證報告的大小；仔細地討論驗證報告在廣播排程中的擺放位置對資料取得所需時間的影響。我們發現將驗證報告擺在每個廣播資料區塊中第一個異動資料之前，可以大量降低客戶端取得所需資料的等待時間。

關鍵詞：資料廣播、快取資料驗證、資料驗證報告

1. 緒論

由於資訊科技的進步，無線網路已經進入成熟的階段，並且改變著人類的生活方式。透過無線網路，人們將可以隨時隨地透過無線網路取得所需要的資訊。在這種情形下衍生出來的運算模式，被稱做**行動計算**模式。所謂行動計算，是指行動客戶端在移動式的環境中移動並且透過無線網路來對於伺服器端提出資料與運算要求的行為。而這種運算環境具有一些限制。在此種新的運算環境之中，引發了許多新的應用方向，而資料散佈應用便是其中的一種。資料散佈的應用十分的廣泛，例如：天氣預報資訊系統，即時交通情報系統，股市交易資訊系統等等，都是標準的資料散佈的應用。在資料散佈系統中，伺服器是唯一的資料來源。由於無線網路上傳與下載頻寬的不對稱性，伺服器端數量與客戶端數量的不對稱性，服務負擔的不對稱性，資料容量的不對稱性…等等的不對稱性使得資料散佈應用的特性與傳統 client/sever 運用的特性並不相同。

資料廣播技術（Data Broadcasting）是一種廣泛被用來解決在資料散佈系統中所遇到的問題。在資料廣播技術中，伺服器端採用廣播的方式，將客戶端需求的資料項目廣播至廣播頻道中。而客戶端僅需要到廣播頻道中等待所需資料的播放。使用資料廣播的方式，伺服器端將可以同時服務大量的客戶端，因此資料廣播技術將可以使伺服器端有效的利用有限的頻寬來滿足大量客戶端的需求。

在廣播方法中，有兩個評估效能的指標-調校時間(tuning time)和獲取時間(access time)。調校時間是指客戶端進入廣播頻頻道中實際下載的封包個數。由於客戶端在存取資料的過程中，客戶端進入廣播頻道中下載資料項是最耗費能源的，因此我們可以利用客戶端取得的封包個數來作為客戶端能源消耗的依據。而獲取時間則是指客戶端提出資料需求到最後取得資料，這段時間伺服端廣播的封包個數。因此獲取時間可以稱為客戶端要拿資料平均所花費的等待時間。

在[2][3]這兩篇論文中曾提到，到廣播頻道中取

得資料通常需要耗費大量的調校時間(tuning time)與獲取時間(access time)，為了增加效率，客戶端多半快取(cache)以往常用的資料項以增加效率。在這兩篇論文中是假設資料項是沒有異動的情形。而在本篇論文中我們是假設資料項是會有異動的情形。因此客戶端快取中的資料項內容有可能被伺服端異動過，導致快取中的資料項失效而無法使用。所以當客戶端需要利用快取中的資料項時，客戶端需要一個方法來確認快取中的資料項是否還可以使用。為了解決這個問題，我們在廣播方法中加入[4]驗證報告(invalidation report, 簡稱 IR)來解決客戶端無法確認快取中資料項是否有被異動的問題。

使用驗證報告的原因是因為在傳統的驗證報告環境中，伺服端是週期性地將驗證報告廣播出去，和廣播索引、廣播資料一樣也是透過"廣播"來將訊息傳達給客戶端。因此在廣播方法中加入驗證報告來解決快取資料的一致性問題是一個十分合理的作法。但是將驗證報告加入廣播方法中仍然有許多問題，我們將在下一章中討論驗證報告加入廣播方法中產生的問題並提出有效的解決方式。

2. 廣播環境下的快取驗證方法

2.1 廣播環境

在廣播結構中，廣播循環是由資料項和索引組成。而索引是樹狀的架構(索引樹)，索引樹的最下層為資料項，資料項的上層為索引。當客戶端需要資料項時，客戶端將會進入廣播頻道中下載根索引。下載根索引後，根據根索引的內容決定下一個所要下載的索引。下載索引後，再根據下載的索引決定下一個索引，反覆地執行相同的步驟，直到客戶端取得需要的資料項。我們利用一個實際的例子來說明。

在圖 1 中，有一索引樹。若客戶端需要資料 5，客戶下載根索引 R，根據根索引 R 的內容決定下載索引 a2。再根據索引 a2 的內容，即可得到資料 5。

由於索引樹是一個二維的架構，因此我們需要一個廣播排程來將索引樹轉換成一維的架構，以便在廣播頻道中一個一個播出。在本篇論文中我們使用在[1]中提過的(1,m) indexing 這個廣播排程，(1,m) indexing 容易實做，且效率能被大家所接受。(1,m) indexing 是將所有的資料項分成 M 塊資料區塊(data segment)，並在這 m 塊資料區塊前面加入所有的索引(即整棵索引樹)

Example 1: 圖 1 為有 9 個資料項的索引樹，當使用(1,3) indexing 時，來產生廣播結構，首先將資料分成 3 塊資料區塊，每塊資料區塊有 3 個資料項(1~3、4～6、7~9)， 並在每塊資料區塊前加入所有的索引 $(R,a_1,a_2\ldots b_9)$，圖 2 為轉換後的廣播結構。

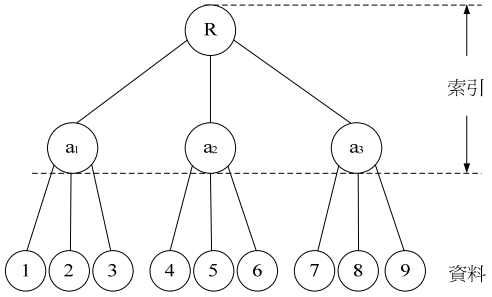

圖 1 索引樹

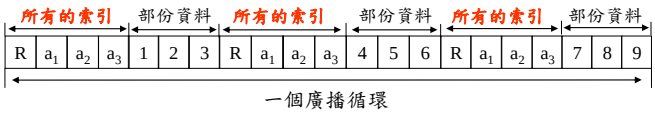

圖 2 廣播排程

為了讓客戶端可以正確地取得需要的資料項，在廣播結構中的索引內容會表示成(key，offset)。key 是用來決定此索引所負責的範圍，當客戶端使用該索引時，客戶端會拿資料項的編號和 key 作比對決定下一個所要拿的封包。offset 則是用來告知客戶端下一個封包還需要多久才會被伺服端廣播，在下一個封包廣播之前客戶端會進入休眠模式，直到下一個封包被廣播才醒來取得封包以節省能源。

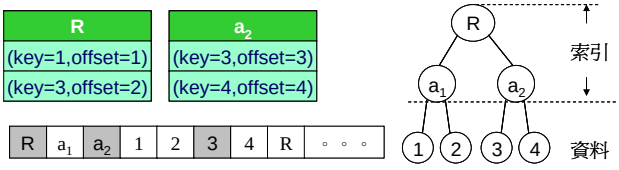

圖 3 索引的內容與使用

Example 2: 在圖 3，當客戶端需要資料項時，客戶端

進入廣播頻道中取得索引(R)，索引(R)有兩個 key 分別為(1,3)也就是說，若需要的資料項編號介於 1~2，客戶端下一個所要拿的封包為索引 a_1，客戶端可以藉由 offset 即可知道索引 a_1 何時才會被伺服端廣播，在此例子 offset 為 1;當資料項編號介於 3~4 時，客戶端下一個所要拿的封包為索引 a_2，客戶藉由 offset 即可知道索引 a_2 何時才會被伺服端廣播，在此例子 offset 為 2。

2.2 廣播環境下的驗證報告

2.2.1 修正封包的表頭

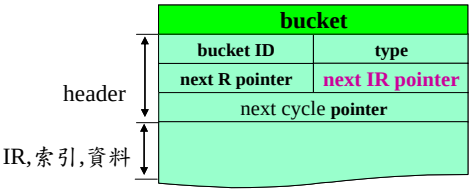

圖 4 封包的表頭

一個廣播循環是由許多的封包(bucket)組成的，封包裡面的內容有可能為索引、資料項、或驗證報告。我們會在每一個封包中加入表頭資訊(header)，圖 4 為封包的表頭資訊，客戶端取得封包後可以從表頭資訊得知下列訊息:

bucket ID :每一個封包在廣播循環中的編號。
Type: 這個封包的內容型態，有三種型態索引、資料和驗證報告。
next cycle pointer: 目前封包到下一個廣播循環開始的距離。
next R pointer: 目前封包到下一個根索引(R)封包的距離。

為了要讓客戶端進入廣播頻道中能順利取得驗證報告，我們在表頭資中新增此欄位。

next IR pointer: 使得客戶端收下封包後，能知道目前封包到下一個驗證報告的距離。

2.2.2 驗證報告的播放和廣播結構的配合

為了要將驗證報告放入廣播結構之中，我們得先決定驗證報告何時會被伺服端廣播，有兩種可能:

1. 和傳統的驗證報告一樣，每隔固定的一段時間就廣播一次驗證報告。

2. 驗證報告配合著廣播結構播出，即驗證報告固定在廣播循環中的某一位置。

在方法 1 中，伺服端每隔固定的封包個數就廣播一次驗證報告，由於每一個廣播循環的長度增加或減少，驗證報告可能會擺在索引之中或資料之中等位置，造成驗證報告擺放的位置不固定，而導致客戶端的獲取時間會增加，所以本篇論文不採取此法。

在方法 2 中，驗證報告擺在廣播結構中固定的位置(ex.驗證報告擺在索引之前)。驗證報告不會因為廣播循環長度的增減，而導致驗證報告擺放的位置每次都不一樣。因此使用本方法可以使得存取驗證報告的步驟較為規律，減少客戶端存取資料的獲取時間，所以在本篇中我們採取驗證報告配合著廣播結構播出。

2.2.3 驗證報告的更新時間

在廣播方法中，伺服端會在廣播循環開始前決定這個廣播循環所要廣播的索引和資料項的內容，一旦決定了內容後，伺服端就不會更改這個廣播循環的索引和資料項內容。若廣播循環中的資料項有被異動時，異動的資料項不會反應在這個廣播循環，而會反應在下一個廣播循環之中，所以驗證報告的更新時間為一個廣播循環。因此在一個廣播循環中放入多個驗證報告，這些驗證報告的內容都會是一樣的，並不會因為在廣播循環中資料項被異動，而改變驗證報告的內容。

由於索引、資料和驗證報告的更新時間為一個廣播循環，廣播循環中資料項的異動會反應在下一個廣播循環中，所以在同一個廣播循環異動資料項的時間戳記，可以用一個編號來代表所有的時間戳記，我們替每個廣播循環編一個對應的編號 cycle number(簡稱 cycle no.)，並用 cycle no.取代驗證報告中的時間戳記。

我們使用一個例子來說明如何製作驗證報告。

Example 3: 在圖 5 中，伺服端在 cycle no.=8 的廣播循環開始前，伺服端要製作驗證報告。假設這個驗證報告包含過去三個廣播循環(cycle no.=5、cycle no.=6、cycle no.=7)的資料異動，將各個廣播循環中異動資料項的編號加入到對應的 cycle no.中。在此例中，在 cycle no.=7 的廣播循環中資料 5 和資料 6 有被異動，將資料 5 和資料 6 加入驗證報告中，並利用 cycle no.當作資料 5 和資料 6 的時間戳記。再將 cycle no.=6 和 cycle no.=5 廣播循環中有異動資料項分別加入驗證報告，如此即可產生出下一個廣播循環的驗證報告。

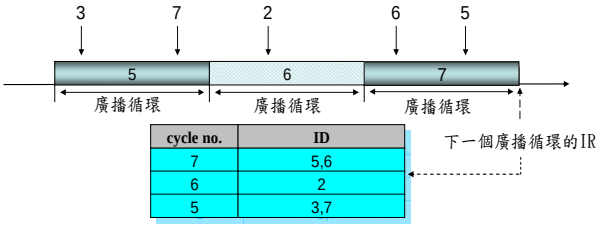

圖 5 廣播環境的驗證報告

2.2.4 驗證報告所擺放位置的最佳化

在廣播方法中，將驗證報告擺在最直觀的位置-每塊索引區塊之前加入驗證報告，所以每一個廣播循環會有三個驗證報告，而且這三個驗證報告的內容會是一樣的。目前的廣播循環如圖 6 所示。

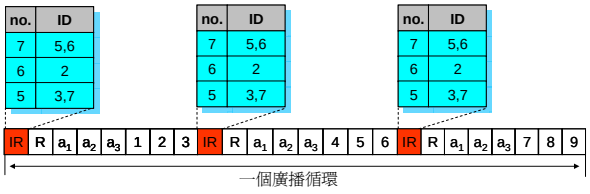

圖 6 廣播循環

在知道目前的廣播循環後，接下來我們介紹客戶端目前的存取程序。圖 7 為客戶端的存取流程，客戶端進入廣播頻道中收下任一封包，利用封包的表頭得知驗證報告還要多久才會被伺服端廣播，客戶端進入休眠模式，直到驗證報告被廣播才醒來取得驗證報告並驗證快取中的資料項，驗證後若需要的資料項還在快取中，客戶端直接利用快取中的資料項。反之，需要的資料項不在快取中，客戶端進入廣播頻道中取得索引，利用索引取得客戶端需要的資料項。

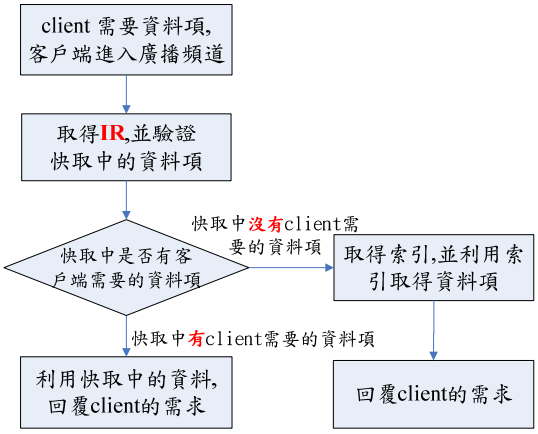

圖 7 存取流程

Example 4 :在圖 8 中，假設客戶端需要資料項 7，且客戶端的快取中有資料項 7，資料 7 的 cycle no.= 4。客戶端進入廣播頻道中取得驗證報告並驗證快取中的資料項，資料 7 的 cycle no.小於驗證報告中資料 7 的 cycle no.(4 < 5)，將資料 7 從快取中刪除。由於客戶端的快取中沒有資料 7，所以客戶端進入廣播頻道中取得索引並利用索引取得資料 7。

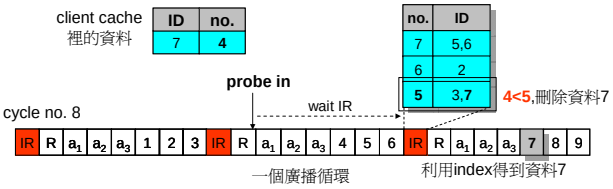

圖 8 在廣播循環中使用驗證報告

當客戶端有快取資料項時，有可能會發生客戶端為了要取得驗證報告而錯過資料廣播。在圖 9 中，假設客戶端需要資料 5 且客戶端有快取資料 5，快取中資料 5 的 cycle no.= 5。客戶端進入廣播頻道中取得驗證報告並驗證客戶端的快取，快取中資料項的 cycle no.小於驗證報告中的 cycle no.(5 < 7)，表示資料 5 是失效的，客戶端將資料 5 從快取中刪除，而客戶端得到下一個廣播循環才可以拿到資料 5。但是在客戶端一開始進入廣播頻道時，資料 5 還沒有被伺服端廣播。而客戶端為了驗證快取中的資料項，反而造成錯失了伺服端廣播資料 5。所以當客戶端有快取資料項時，有可能會發生客戶端為了要取得驗證報告而錯過資料廣

播。而造成客戶端的獲取時間大幅地提升。

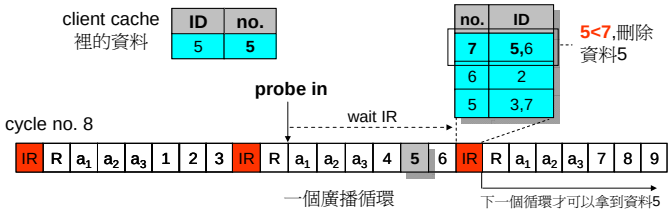

圖 9 驗證資料所造成的問題.

為了減少客戶端發生錯失資料廣播的情形，我們希望驗證報告擺放的位置和資料區塊越接近越好，因此我們將驗證報告擺放的位置往後移，驗證報告可能的擺放位置有以下三種:

1.驗證報告擺在索引之前

2.驗證報告擺在索引之後資料之前

3.驗證報告擺在資料之中

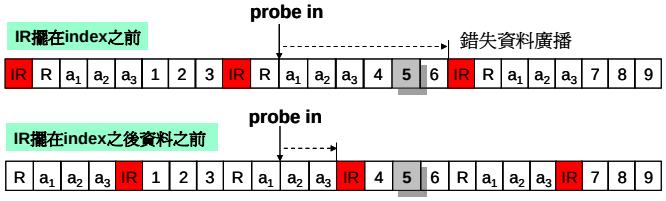

圖 10 比較驗證報告擺放的位置

首先我們先討論前兩種驗證報告擺放的位置。在圖 10 中，有兩個不同的廣播循環，一種為驗證報告擺在索引之前，另一種為驗證報告擺在索引之後資料之前。假設客戶端需要資料 5 且快取中有資料 5，客戶端在相同的時刻 probe in。當驗證報告擺在索引之前時，客戶端會因為接收驗證報告，而造成錯失伺服端廣播資料 5。當驗證報告擺在索引之後資料之前時，客戶端接收完驗證報告，資料 5 還沒被伺服端廣播。相較於驗證報告擺在索引之前，驗證報告擺在索引之後資料之前較能減少客戶端為了接收驗證報告，而造成錯失資料廣播的情形。

驗證報告擺在索引之後資料之前會遇到一個問題。當客戶端取得驗證報告後會因為要取得索引而錯過資料廣播。在圖 11 中，假設客戶端需要資料 5 且快取中有資料 5。客戶端進入廣播頻道中取得驗證報告，驗證完快取中的資料項後，雖然資料 5 在驗證報告之後的資料區塊中，客戶端還是必須取得索引才能取得資料，如此又會造成客戶端錯失了資料的廣播。為了解決上述的問題，我們在驗證報告中加入 offset，offset 的內容為資料項的廣播時間。若在驗證報告中出現的資料項編號，在驗證報告之後的資料區塊中有被伺服端廣播，則將資料項所對應的 offset 加入驗證報告中。我們使用一個例子說明。

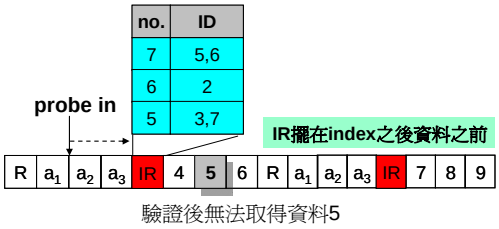

圖 11 驗證報告擺在索引之後資料之前

Example 5: 在圖 12 中，在驗證報告中有包含資料 5 和資料 6，且在驗證報告厚的下一塊資料區塊中有廣播資料 5 和資料 6，伺服端將資料 5 和資料 6 對應的 offset 分別加入驗證報告中，在此例中，資料 5 的 offset=2，資料 6 的 offset=3。

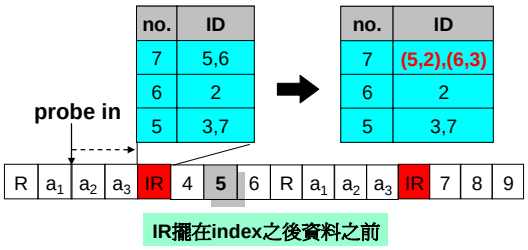

圖 12 在驗證報告中加入 offset

將 offset 加入驗證報告後，可以使得客戶端取得驗證報告驗證後，還有機會取得下一個資料區塊中的資料項。我們用一個簡單的例子說明客戶端如何利用 offset 取得資料項。

Example 6: 在圖 13 中，假設客戶端需要資料 5，且快取中有資料 5，客戶端收到驗證報告並驗證資料 5，快取中資料項的 cycle no.小於驗證報告中的 cycle no.(5<7)，將資料 5 從快取中刪除。客戶端在驗證報告

中找到資料 5 的 offset，在此例資料 5 的 offset=2，客戶端由 offset 即可知道資料 5 廣播的時間，客戶端進入休眠，直到資料 5 廣播時醒來取得資料 5。

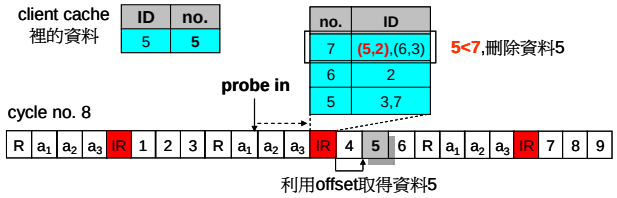

圖 13 客戶端存取資料

客戶端驗證快取中的資料，有可能會造成錯失資料廣播的情形，為了要減少這種情形，驗證報告擺放的位置和資料越接近越好，所以我們將驗證報告擺放的位置從索引之後資料之前移到第一個異動資料之前。如圖 14 所示，我們將驗證報告擺在第一個異動資料之前，以資料區塊(1、2、3)為例，第一個異動的資料為資料 2，所以我們將驗證報告從索引之後資料之前往後移到資料 2 之前，以達到減少錯失資料廣播的情形。

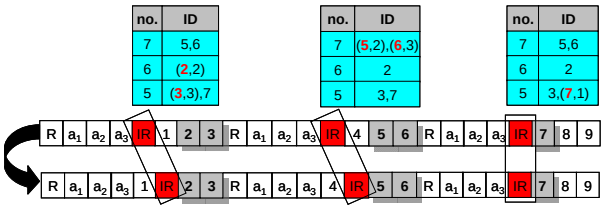

圖 14 將驗證報告擺在第一個異動的資料項之前

2.2.5 客戶端的存取程序

介紹完驗證報告擺放的位置後，我們將介紹客戶端的存取程序，當客戶端需要資料時，分為兩種不同的狀況；一種為客戶端有快取所需的資料項（如圖 15），另一種為客戶端沒有快取所需的資料項(如圖 16)，面對這兩種不同的情況，客戶端採取不同的存取程序。

圖 15 為客戶端有快取所需要的資料項的存取流程。客戶端希望需接收驗證報告來確認快取中的資料項是否可以使用，客戶端進入廣播頻道中接收任一封包，知道驗證報告何時廣播，等到驗證報告廣播時收下驗證報告並驗證客戶端的資料，若需要的資料有在快取中，則利用快取中的資料項回覆客戶端的需要。反之，若需要的資料項沒有在快取中，則客戶端從取得的驗證報告中尋找有無所需資料對應的 offset，若有對應的 offset，則利用該 offset 取得資料；若找不到 offset，則到廣播頻道中取得索引，並利用索引取得資料項。

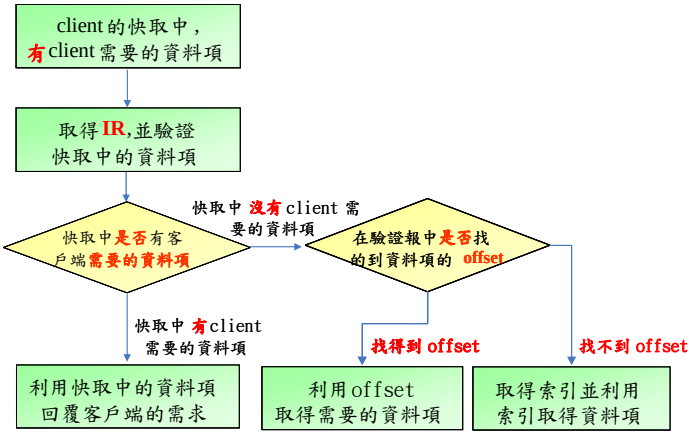

圖 15 客戶端存取流程

圖 16 為客戶端沒有快取所需資料的存取流程。客戶端進入廣播頻道中取得需要的資料項，但是由於快取中還有其他的資料項，為了這些資料項的一致性，客戶端仍然需要到廣播頻道上取得驗證報告，所以當客戶端進入廣播頻道中接收任一封包時，藉由封包表頭的 Next_IR_pointer 和 Next_R_pointer 欄位來決定要先存取驗證報告還是先取得資料，若 Next_IR_pointer < Next_R_pointer，表示驗證報告比索引先被廣播，則先取得驗證報告，驗證快取中的資料項，若 IR 中有所需資料的 offset，則以之取得資料；否則，取得索引 R 並藉由索引取得需要的資料。反之，若 Next_IR_pointer > Next_R_pointer，表示索引比驗證報告先被廣播，則客戶端先取得索引 R，再取得所需資料及驗證報告。

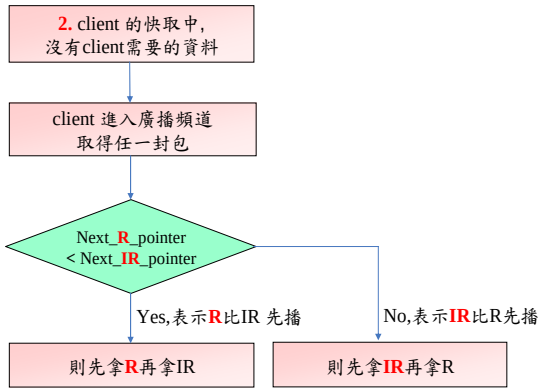

圖 16 客戶端存取流程

3. 實驗結果與分析

我們會介紹本實驗的參數所代表的意義和參數的設定值。並由我們由實驗的結果來分析方法的優缺點。

3.1 實驗參數的介紹

表 1 參數設定

項目	值
m	2
索引樹的分支度	4
索引樹的階度	6
資料項的個數	4096(bucket)
快取空間	20%~100%
平均資料異動的時間	0.5~0.001

實驗的參數設定如表 1 中所示,在本實驗中的索引樹的分支度為 4,索引樹的階度為 6,索引樹的最下層為為資料項,資料項總共有 4096 個。

我們的廣播排程是根據[1]中的(1，m)indexing 廣播排程。而在[1]中曾提到 m 值的最佳化如下所示:

$$m = \sqrt{\frac{Data}{Index}}$$

Data 為索引樹中資料項的個數，Index 為索引樹中索引的個數，在本實驗中資料項的個數為 4096 個，而索引的個數為 1366，所以我們採用(1,2)indexing 的廣播排程。

在實驗中,我們假設資料項的異動時間的間隔服從指數分配，並以 T_{DUI} 表示該指數分配的期望值，若 u 表示資料異動的間隔時間，則間隔時間的分配如下:

$$P(u) = \begin{cases} \frac{1}{T_{DUI}} e^{\frac{-u}{T_{DUI}}} & ,u > 0, \\ 0 & ,\text{otherwise.} \end{cases}$$

其中 $E(u)= T_{DUI}$。

3.2 實驗結果

在本節中，3.2.1 中，我們會介紹本實驗評估效能的標準，接下來我們會介紹本實驗的對照組。在 3.2.2 中我們會針對實驗的結果提出說明與分析。

3.2.1 實驗簡介

在我們的實驗中,我們以調校時間與獲取時間作為評估效能的標準。調校時間為客戶端進入廣播頻道中下載的封包個數，單位為封包個數(bucket)。獲取時間為客戶端取得資料平均所花的等待時間，單位為封包個數(bucket)。

在本實驗中，有兩個主要的對照組。首先我們先介紹第一個對照組，我們在實驗中稱此對照組為“No-cache”。在 No-cache 中，我們假設客戶端沒有快取空間，因此當客戶端需要資料項時，客戶端進入廣播頻道中取得索引，並藉由索引取得資料項。

再來我們介紹另一個對照組，我們在實驗中稱此對照組為“IR- before-index”。在 IR-before-index 中，假設客戶端有快取空間，且伺服端廣播的驗證報告擺在索引之前。當客戶端需要資料項時，客戶端得先取得驗證報告並驗證快取中的資料，若資料項有在快取中，則使用快取中的資料項。反之，客戶端進入廣播頻道中取得索引，並藉由索引取得資料項。

3.2.2 實驗數據分析

在這一節中，我們會調整參數的大小，觀察調整參數的大小對每個方法的影響程度。

實驗一:資料異動的間隔時間對效能的影響

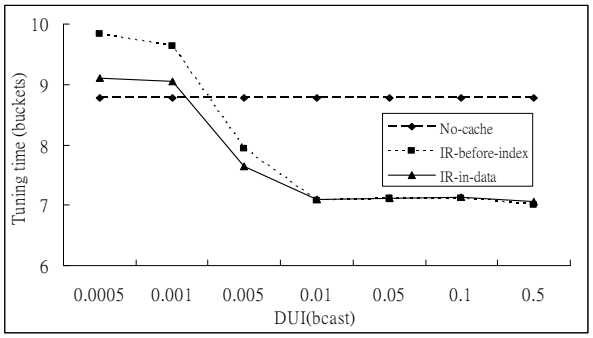

圖 17 資料異動間隔時間的大小對調校時間的影響

在圖 17 中，圖的 X 軸是 T_{DUI}，T_{DUI} 是指平均資料異動的間隔時間，單位為廣播週期。圖的 Y 軸是調校時間，調校時間是指客戶端下載的封包個數，單位為封包個數。在圖 17 中，我們比較了三種方法 -No-cache、IR-before-index 和我們的方法 IR-in-data(驗證報告擺在第一個異動資料之前)。在此實驗中，IR-before-index 和 IR-in-data 兩個方法的快取空間為 40%，即客戶端可以快取 40%的資料項。

在圖 17 中，我們可以看出 T_{DUI} 對調校時間的影響。當 T_{DUI} 越大，表示資料異動的間隔時間越大，也就是資料異動的頻率越小。首先我們先比較有快取空間和沒快取空間的調校時間。由圖 17 可以看出 T_{DUI} 的大小對於 No-cache 的調校時間不會有影響，主要是因為客戶端沒有快取資料，所以資料的異動並不會影響客戶端存取資料所耗的調校時間。而 IR-before-index 和 IR-in-data 這兩個方法均有快取空間，兩個方法在 T_{DUI}=0.001，兩個方法所花費的調校時間都會比 No- cache 來得大。隨著 T_{DUI} 的增加，兩者的調校時間會越來越低，且其調校時間都會明顯的比 No-cache 來得小。雖然這兩個方法的變化方向都一致，但是當 T_{DUI} 值越小時 IR-in-data 的調校時間會比 IR-before-index 來得低。主要是因為在 IR-in-data 這方法中，當客戶端驗證完快取後，若客戶端需要的資料項不在快取中時，客戶端有機會從驗證報告中找到資料項的 offset，並藉由 offset 取得資料項，因此省下了下載索引所耗費的調校時間，所以 IR-in-data 的調校時間會比 IR-before-index 來得低。

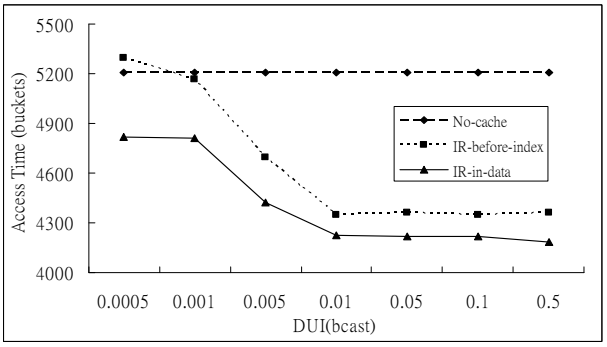

圖 18 資料異動間隔時間的大小對獲取時間的影響

在圖 18 中，圖的 X 軸是 T_{DUI}。圖的 Y 軸是獲取時間，獲取時間是指客戶端取得資料平均所花的等待時間，單位為封包個數。假設快取空間為 40%，即客戶端可以快取 40%的資料項。

首先我們比較有快取和沒快取對獲取時間的影響。由圖 18 可以看出 T_{DUI} 值的變化對於 No-cache 的獲取時間不會有影響，主要是因為客戶端沒有快取資料，所以資料的異動並不會影響客戶端存取資料所耗的獲校時間。另外兩個方法由於擁有快取空間，所以一旦快取命中，則客戶端可以從快取中直接取得資料，因此節省大量的獲取時間。所以另外兩個方法的獲取時間會比 No-cache 來得小。T_{DUI} 的值較大，代表資料異動越不頻繁，導致快取命中(cache hit)的機率較高，所以獲取時間也較短。如圖 18 中所示， IR-in-data 的獲取時間較 IR-before-index 的獲取時間來得短。主要是因為 IR-in-data 方法中，IR 所放的位置可以儘量減少客戶端為了取得驗證報告而錯失資料廣播。而當 T_{DUI} 很小時，代表資料異動非常頻繁；此時，IR-before-index 及 IR-in-data 方法下，客戶端所快取的資料項皆失效；但在 IR-in-data 的方法下，由於客戶端可以利用 IR 中 offset 欄位取得所需資料，因此獲取時間較低。

實驗二:快取空間的大小對效能的影響

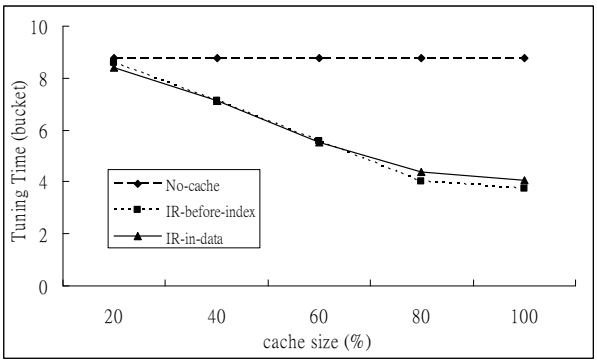

圖 19 快取空間的大小對調校時間的影響

在圖 19 中，圖的 X 軸是客戶端的快取空間，單位為%。圖的 Y 軸是調校時間。在此實驗中的資料異動間隔 $T_{DUI} = 0.05$。

在圖 19 中，我們可以看出快取空間和調校時間的關係。由圖可以看出當快取空間逐漸增加，會造成客戶端快取命中(cache hit)的機率提升，所以 IR-in-data 和 IR-before-index 的調校時間會明顯的變低。

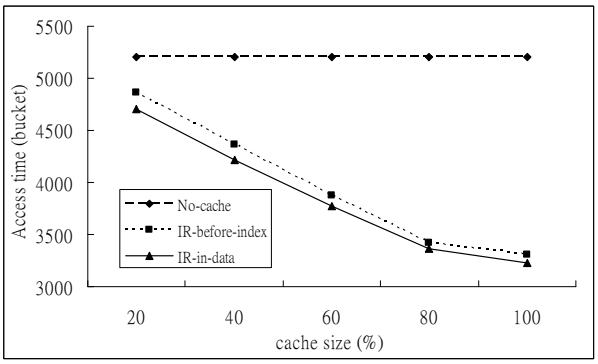

圖 20 快取空間的大小對獲取時間的影響

在圖 20 中，圖的 X 軸是客戶端的快取空間，單位為%，圖的 Y 軸是獲取時間，單位為封包個數。在此實驗中的資料異動間隔 T_{DUI} 為 0.05。

在圖 20 中，我們可以看出快取空間和獲取時間的關係。當快取空間逐漸增加，會造成客戶端快取命中的機率提升。如此會導致客戶端的獲取時間減小。由此圖可以發現 IR-in-data 和 IR-before-index 相比，IR-in-data 的獲取時間較短，這是因為 IR-in-data 可以儘量減少客戶端為了取得驗證報告而錯失資料廣播的情形。由圖 19 及圖 20 所示，適當的資料快取，對於客戶端的調校時間及資料獲取時間有顯著的好處。

4. 結論

在資料廣播的環境中作資料快取，會遇到許多的問題。在本篇論文中，我們將資驗證報告(IR)加入廣播結構中，以維持客戶端快取資料的一致性。我們修改了封包的表頭，使得客戶端可以得知驗證報告廣播的時間。並且根據廣播環境的特性，使用廣播編號(cycle no.)來取代驗證報告中的時間戳記，以降低驗證報告所佔的空間。此外，由於驗證報告擺放的位置會影響客戶端的獲取時間，本論文比較兩種驗證報告擺放位置對客戶端取得資料所需的等待時間的影響。IR-before-index是將IR擺在每個索引區塊之後，而IR-in-data是將IR擺在每個資料區塊中，第一個被異動資料之前；經實驗發現，IR-in-data和IR-before-index在資料異動的頻率小時，可有效地降低調校時間和查詢等待。而IR-in-data所需的查詢資料等待時間比IR-before-index來得低。

致謝：本研究由國科會計畫編號 NSC 95-2221-E-011-070- 支助完成，謹此致謝。

參考文獻

[1] T. Imielinski , S. Viswanathan, and B. R. Badrinath, “Data on Air: Organization and Access,” IEEE Trans. Knowledge and Data Engineering, Vol. 9,No. 3, pp353-372 , May/June 1999.

[2] D.A. Tran, K.A. Hua, and K. Prabhakara, “On the Efficient Use of Multiple Physical-Channel Air-Cache,” Proc. IEEE Wireless Communications. and Networking Conf. (WCNC ’02), Mar. 2002.

[3] S. Acharya et al., “Broadcast Disks: Data Management for Asymmetric Communications Environments,” Proc. ACM SIGMOD Conf., pp. 199-210, May 1995.

[4] D. Barbara and T. Imielinski. “Sleepers and Workaholics: Caching Strategies in Mobile Environments.” In Proceedings of the ACM SIGMOD Conference on Management of Data, pages 1-12, 1994.

[5] C.J. Su and L. Tassiulas, “Joint broadcast scheduling and user’s cache management for efficient information delivery,” ACM Wireless Networks, 6, 2000.

[6] S. Acharya et al., “Prefetching from a Broadcast Disk,”*12th Int’l Conf. Data* Eng., pp. 276-285, Feb. 1996.

[7] S. Hameed and N.Vaidya, “ Efficient algorithm for scheduling data broadcast Wireless Networks,” 5:183-193, 1999.