

A Practical Approach of Composing Near-Optimal Invalidation Reports for Mobile Databases

Wen-Chi Hou, Chih-Fang Wang, Haiqing Li

Department of Computer Science, Southern Illinois University, Carbondale IL 62901
{hou, cfw}@cs.siu.edu

Abstract

Caching can reduce expensive data transfers and improve the performance of mobile computing. To reuse caches after short disconnections, invalidation reports are broadcast to clients to identify outdated items. Detailed reports may not be desirable because they can consume too much bandwidth. On the other hand, false invalidations set in if accurate timing of updates is not provided. We observe that false invalidation rates are closely related to clients' reconnection patterns and show that in theory, for any given reconnection pattern, a report with a minimal false invalidation rate can be derived. For practical uses, we capture the reconnection pattern by sampling and use the pattern to compose a near-optimal invalidation reports. This method is simple and fast. Experimental results have confirmed that our method is indeed close to optimal and more effective in reducing the false invalidation rate than others.

Keyword: Invalidation report, Mobile database, False invalidation, Caching.

1. Introduction

Without wires to impede mobility, today's cordless and cellular telephones provide the ultimate accessibility, convenience, and flexibility. Countless mobile databases are in use today, including mobile surgical databases [24], stock information systems, and a variety of general purposed pocket databases for small business and personal use. In a mobile or nomadic computing environment [5], a set of database servers disseminates information via wireless channels to mobile clients [25]. This environment no longer requires clients to stay in fixed positions in the network. Clients can relocate and connect to different database servers at different times.

Due to the narrow bandwidth of wireless channels, communication between the clients and servers ought to be minimized to reduce contentions. Caching of frequently accessed data at mobile clients has been shown to be a very effective mechanism in handling this problem. Other benefits of caching include energy savings (by reducing the amount of data transferred) and cost savings (especially if one is billed on a pay-per-packet basis).

In general, updates to database are broadcast by the server without much delay to the clients. Therefore, as long as the clients stay connected, their caches are current. However, in a mobile database environment, clients are frequently disconnected due to battery power saving measures [12] or unpredictable failures. Discarding entire caches after short disconnections could be wasteful as many data items in the caches may still be valid; this is especially true for a mobile database environment, where the bandwidth is narrow and battery power (of clients) is limited. Therefore, an efficient scheme for reuse of cache is much needed [21].

Many caching coherence algorithms have been proposed for conventional client-server architectures [7, 10, 11, 26, 28]. However, due to the unique features of the mobile computing environment, such as narrow bandwidth, frequent disconnections, weak communication capability (of clients), etc., these algorithms may not be directly applicable. There have been some researches on cache management for mobile computing published recently in the literature [2, 6, 8, 9, 18, 19, 21, 22, 29 etc.]. To reuse the caches after short connections, a common approach is to broadcast invalidation reports to clients to help identifying outdated items in the caches [3, 14, 16]. Detailed reports can be long, consuming much bandwidth and thus may not be desirable. On the other hand, without detailed timing information, cached items can be falsely invalidated. In this paper, we discuss how to compose a report with a minimal false invalidation rate. We observe that false invalidation rates have to do with clients' reconnection patterns, that is, the distribution of the time spans between disconnection and reconnection. By applying Newton's method to the clients' reconnection pattern, we show that a report with a minimal false invalidation rate can be derived. For practical uses, we devise a method that yields a near-optimal invalidation report that can be composed efficiently. Simulation results have also confirmed that our near-optimal method is fast and effective in reducing false invalidations.

The rest of the paper is organized as follows. In Section 2, we review the mobile computing environment and research on cache management for such an environment. Section 3 reviews the compositions of invalidation reports to lay down the groundwork for our approach. In Section 4, we take clients' reconnection patterns into account in the design of invalidation reports. We show that by using Newton's method, reports with minimal false invalidation rate can be obtained. In Section 5, we compose a near-optimal report that can be efficiently composed. Simulation results are presented in Section 6, and we conclude our study in Section 7.

2. Preliminary

2.1 A Mobil Client-Server Computing Architecture

Today, most commercial relational database management systems are based on the client-server architectures. In the client-server architecture, the database resides at the server and is accessed by application programs running on the clients' workstations. In a mobile client/server computing environment, clients no longer are required to remain in fixed positions. Figure 1 shows a general architecture for a mobile client-server computing paradigm that is similar to [15].

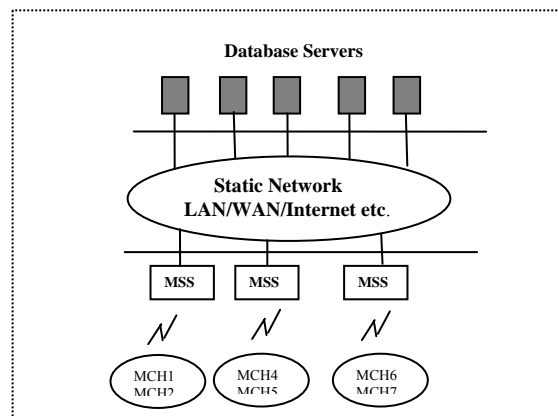


Figure 1. A Mobile Computing Architecture

It consists of a large number of mobile client hosts (MCHs) and relatively fewer, but more powerful fixed hosts that are connected through a wired network. MCHs usually have weaker transmitting capability than fixed hosts. Some of the fixed hosts, called the MSS (Mobile Support Stations) [4], are equipped with a wireless interface to communicate with MCHs. Fixed hosts can communicate with other hosts (mobile or fixed) via a wireless channel.

An MCH can directly communicate with an MSS if it is physically located within the cell serviced by the MSS. A wireless communication channel comprises an uplink (from a client to the server) and a downlink (from the server to a client) sub channels. An MCH submits requests to the local MSS via an uplink channel and receives the results via a downlink channel. Due to the weaker transmitting capability of the mobile clients (as compared to the fixed hosts), an asymmetric communication with much smaller uplink bandwidth than downlink is created. Receiving messages is less costly than sending messages for clients.

2.2. Cache Management

There have been some researches on designing cache coherence schemes for mobile computing. Refresh time [11] has been used to determine the validity of cached items. Each item cached is associated with a refresh time. When the refresh time expires, the client contacts the server for an updated value. However, appropriate refresh durations are difficult to determine. Dynamic refresh time [10], based on the update frequency of the items, has been proposed as an improvement. However, many cached items can still be falsely invalidated (i.e., valid items considered as invalid) while others may be falsely validated (i.e., invalid items considered as valid). The cache coherency is not maintained effectively and consistently.

The cache invalidation method was proposed [3] to minimize the data transfer in both directions: the downlink and the uplink. In this approach, the server, periodically or asynchronously, broadcasts reports containing items that have been updated. When a client receives such reports, it checks against its cache and invalidate. Due to its simplicity and efficiency, invalidation reports have been adopted in many recent researches [17, 18, 22, 27].

3. Invalidation Reports

Based on the timing of the invalidation messages being broadcast, cache invalidation methods can be either asynchronous or synchronous. In an asynchronous approach [17, 25, 27], a server broadcasts an invalidation message as soon as an item changes its value. One problem with the asynchronous approach is its unpredictable waiting time for such invalidation messages. Some improvements have been made [22] such that at least one broadcast is sent in a given period. In the synchronous approach [3, 25, 28], cache invalidation messages are broadcast periodically. That is, the server gathers updates for a period of time and then broadcasts these updates with the time of updates all in one message. Once invalid items are found in the cache, the client submits an uplink request for updated values. Despite the differences in cache coherence schemes, it is generally assumed that outdated items in caches are updated immediately. Therefore, as long as a client stay connected, its cache remains current.

While invalidation reports can be used to invalidate outdated items in the cache, another use of invalidation reports is to help reconnected users identify outdated items without discarding the cache after short disconnections. Discarding entire caches after short disconnections can be wasteful, as many data items in the caches may still be valid. This is particularly important to the mobile databases, where the bandwidth is narrow and battery power (of clients) is limited [5, 21].

In the following sections, we will focus on composing invalidation reports for reconnected clients. We first briefly describe existing methods for composing invalidation reports.

3.1 Broadcasting Timestamp (BT) Strategy

Broadcasting timestamp (BT) strategy [3] was developed based on the synchronous invalidation approach. The report is composed of a set of pairs (ID, timestamp), in which ID specifies an item that has been updated, and the timestamp indicates when a change was made to that item. Note that a report can only cover the activities of a limited period of time, called a window period. The longer the window period, the larger the invalidation report. If a client has disconnected longer than the window period, the entire cache must be discarded.

3.2 Bit-Sequence (BS) Approach

In the BS approach [16], each data item is mapped to a bit of an N -bit sequence, where N is the total number of data items in the database. That is, the n^{th} bit in the sequence corresponds to the n^{th} data item in the database. A value “1” in a bit indicates the corresponding data item has been changed; and “0” indicates otherwise. It reduces the naming space from $N \log(N)$ bits for N items (in the BT approach) to N bits here. Note that in BT at least $\log(N)$ bits are needed to store the ID of an item.

To further reduce the size of a report, instead of using one timestamp for each updated item, the report uses only one timestamp. A bit value “1” in the bit-sequence now indicates that the corresponding item has been updated since the timestamp of the report. A bit value “0” indicates no change has been made to that item since the timestamp. Unfortunately, this measure also decreases the accuracy of the report since there is only one timestamp in a report. All items mentioned in the report have to be treated as being updated at the time indicated by the timestamp. As a result, valid items could be falsely invalidated; that is, false invalidations set in. For example, consider a client disconnected at time d , as shown in Figure 2. After reconnected, it received a report with a timestamp t . Recall that updates are broadcast by the server and are immediately reflected in clients’ caches. Therefore, the client still has valid copies for those items updated between t and d . Since there is no way to tell these items from items updated after d in the report, all the items mentioned in the report will all have to be purged when the client reconnects.

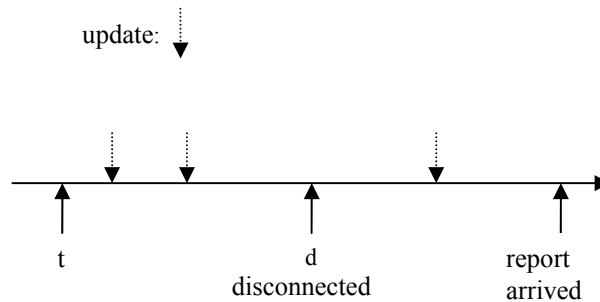


Figure 2: False Invalidations

In order to reduce the false invalidations, a hierarchically structured and more detailed report has been proposed [16]. Instead of using just one bit-sequence (and a timestamp), n bit-sequences ($n > 1$), each is associated with a timestamp, are used in the report to show the update activities of n overlapping subintervals. Specifically, the i^{th} sequence ($0 \leq i \leq n-1$), denoted by B_i , has a length of $N/2^i$ bits, where N is the number of data items in the database, and it records the

latest $N/2^{i+1}$ update activities. Each bit-sequence is associated with a timestamp $T(B_i)$ indicating since when there have been such $N/2^{i+1}$ updates. As shown in Figure 3, the first bit-sequence B_0 has a length of N and half of it are “1” bits, indicating that $N/2$ items (out of N) have been updated since $T(B_0)$. The second sequence B_1 has $N/2$ bits, each corresponding to a data item that has a “1” bits in B_0 . In general, the j^{th} bit of the B_i corresponds to the j^{th} “1” bit in the B_{i-1} , and half of each bit-sequence are 1’s. The total number of bit-sequences n is $\log(N)$.

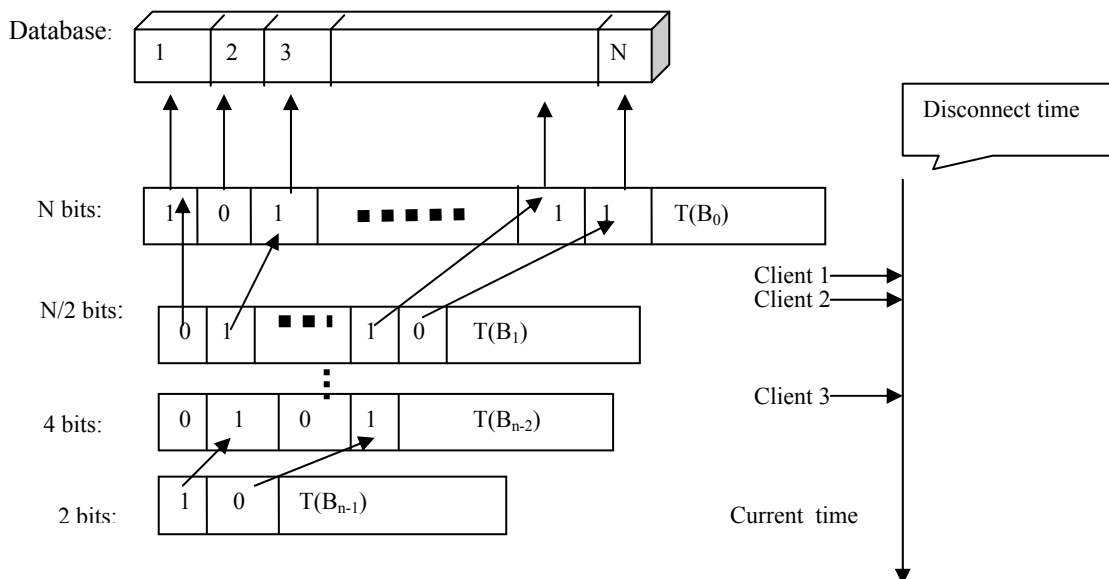


Figure 3: An Invalidation Report with Client Disconnection Time

3.2.1 A Dynamic BS Scheme

The above BS scheme has been modified in [16] to accommodate a variable number of updates (instead of just $N/2$) for the reported period. The modified scheme is called the dynamic BS. Instead of mapping an item to a bit in B_0 , each updated item is represented explicitly by its ID in the dynamic BS. Thus, B_0 is now made of the IDs of those items that have been updated since $T(B_0)$. The rest of bit-sequences (B_1, \dots, B_{n-1}) are composed in the same way as in the original BS. That is, sequence B_i ($0 < i \leq n-1$) has $k/2^{i-1}$ bits, with half of them being “1”s, where k is the number of items updated since $T(B_0)$.

If both an ID and a timestamp are implemented as a 32-bit integer, the total size of a report can be calculated approximately as $32k + 2k + 32 \log(k)$, where $32k$ is the size of k IDs (i.e., B_0), $2k$ is the size of the rest of the bit-sequences (i.e., B_1, \dots, B_{n-1}), and $\log(k)$ is the number of timestamps (or the number of bit sequences) [16].

4. Composition of Optimal Hierarchical Bit-Sequences

Although Jing’s [16] hierarchically structured bit-sequences discussed above reduced the naming space of items and the number of timestamps, there is no justification for why the bit-sequence hierarchy should be partitioned based on half of the updates, i.e., $N/2^i$ or $k/2^i$ updates. In fact, this “half-update-partition” scheme could favor one situation over another. Here, we use Figure 3 again as an example. After reconnecting at (or before) the current time, clients 1 and 2 can use B_0 to invalidate their items, and client 3 will use B_1 . Since clients 1 and 2 disconnected

between $T(B_0)$ and $T(B_1)$, as explained earlier in Figure 2, all cached items updated during this period will have to be invalidated and some of them might be falsely invalidated. Thus, if there are a large number of clients like clients 1 and 2, who disconnected during this (likely long) period, there can be a lot of items falsely invalidated. On the other hand, since there is no client disconnected between $T(B_{n-2})$ and $T(B_{n-1})$, the bit-sequences B_{n-2} is wasted. The above scenario is likely to happen because the time span between $T(B_0)$ and $T(B_1)$ is likely much longer than that between $T(B_{n-2})$ and $T(B_{n-1})$. As observed, there are more bit-sequences covering the more recent but shorter periods than the earlier but longer periods in Jing’s approach. Clearly, this hierarchical structure with “half-update- partition” may not yield minimal false invalidations.

In this research, we attempt to find hierarchically structured bit-sequences that minimize the false invalidation rate. Specifically, we will investigate the division of the n overlapping subintervals in the report such that the false invalidations can be minimized.

The window size of an invalidation report W refers to the time period covered in a report, that is, from $T(B_0)$ to the current time. Here, we assume that W is fixed and predetermined by the system.

4.1 Reconnection Patterns

It is observed that the false invalidations are due to the inaccuracy of the report as the exact time of updates has been replaced by fewer rough timestamps (to save bandwidth). Increasing the number of bit-sequences (and also the timestamps) can certainly increase the accuracy of the report and thus reduce the false invalidation; however, this could also compromise the goal of saving bandwidth. In this research, we consider how to make the best use of bit sequences to lower the false invalidations.

As noted earlier, some of the bit-sequences are of little or no use because there are few or no clients disconnected during that period. Those bit-sequences could have been used more effectively if we had placed them in the periods with more disconnected clients. That is, the effectiveness of the bit-sequences is closely related to the reconnection patterns of mobile clients; i.e., how long clients are likely to reconnect after disconnection. Therefore, to reduce the false invalidation rates, a reorganization of the bit-sequences that takes into account clients’ reconnection patterns needs to be devised.

Assume that the reconnection pattern [20] can be represented by a certain probability distribution such as the one shown in Figure 4, where the X-axis is the time lapsed from the last disconnect time to the reconnect time, while the Y-axis represents the number of clients.

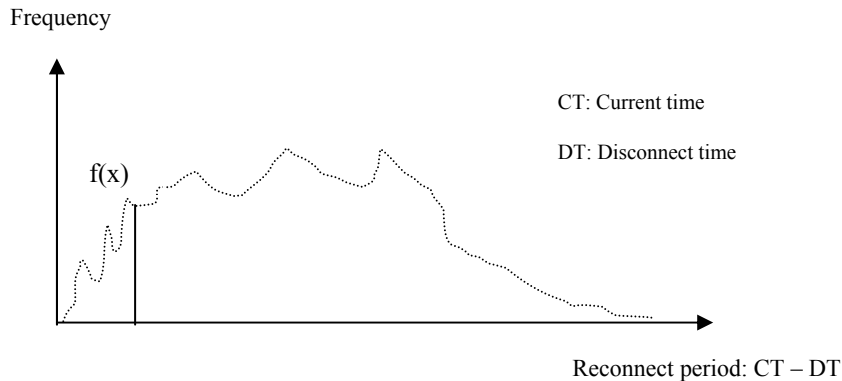


Figure 4: Reconnect Time Distribution

Let us analyze the relationship between the false invalidations and the reconnect distributions. Assume that a mobile client disconnected at time x . After it reconnects and receives its first report, it looks for a sequence, say B_i , in the report with the largest timestamp that is less than or equal to its disconnection time x (i.e., $T(B_i) \leq x$). If the client did not disconnect exactly at $T(B_i)$, there might be a chance for false invalidation, because the client may have already updated some of the items in its cache between $T(B_i)$ and x when it was still connected. The larger the difference between x and $T(B_i)$, the more items might have been updated by the clients before disconnection, and those items would be falsely invalidated when reconnected.

4.2 An Optimal Report with Minimal False Invalidation

Now, let us derive the relationship between the false invalidation and the partition of window for a given reconnection pattern. Since the server receives update requests from users of all kinds, we may assume that updates to the database are independent. Let c be the update arrival rate during the window period. Then, the expected number of items falsely invalidated for a client disconnected at x , denoted $FI(x)$, is $c \cdot (x - T(B_i))$.

Let $f(x)$ be the reconnection pattern. Then, the expected total number of falsely invalidated items, denoted by TFI , is

$$\begin{aligned}
 TFI &= c \sum_{i=0}^{n-1} \left(\int_{T(B_i)}^{T(B_{i+1})} (x - T(B_i)) \cdot f(x) dx \right) \\
 &= c \sum_{i=0}^{n-1} \left(\int_{T(B_i)}^{T(B_{i+1})} x f(x) dx \right) - c \sum_{i=0}^{n-1} \left(\int_{T(B_i)}^{T(B_{i+1})} T(B_i) f(x) dx \right) \\
 &= c \int_{T(B_0)}^{T(B_n)} x f(x) dx - c \sum_{i=0}^{n-1} \left(\int_{T(B_i)}^{T(B_{i+1})} T(B_i) f(x) dx \right) \tag{4.1}
 \end{aligned}$$

where n is the number of bit-sequences in the report, $T(B_n) = CT$ (i.e., the current time). To minimize TFI is equivalent to maximizing $\sum_{i=0}^{n-1} \left(\int_{T(B_i)}^{T(B_{i+1})} T(B_i) f(x) dx \right)$ since $\int_{T(B_0)}^{T(B_n)} x f(x) dx$ is a constant.

The problem becomes how to partition the window into n subintervals to maximize $\sum_{i=0}^{n-1} \left(\int_{T(B_i)}^{T(B_{i+1})} T(B_i) \cdot f(x) dx \right)$. The following theorem [13] justifies the existence of a subdivision that maximizes the formula.

Theorem. Suppose $f(x)$ is a continuous positive real function on interval $[a, b]$, and n is a positive integer. Then, there exists a vector $X = [x_1, \dots, x_{n-1}]^T$ such that $g(X) = \sum_{i=0}^{n-1} \left(\int_{x_i}^{x_{i+1}} x_i f(x) dx \right)$,

where $a = x_0 \leq x_1 \leq \dots \leq x_{n-1} \leq x_n = b$, attains its maximum. Furthermore, the value of X is the solution to the nonlinear system

$$\int_{x_i}^{x_{i+1}} f(x) dx + f(x_i)(x_{i+1} - x_i) = 0, \quad i = 1, 2, \dots, n-1. \tag{4.2}$$

To minimize TFI is equivalent to maximize $\sum_{i=0}^{n-1} \left(\int_{T(B_i)}^{T(B_{i+1})} T(B_i) f(x) dx \right)$. Hence, $T(B_i)$, $i=1, \dots, n-1$ are the solution of (4.2) according to the theorem. Let $F_i(X) = \frac{\partial g(X)}{\partial x_i}$, then Eq. (4.2) is equivalent to

$$F(X)=[F_1(X), F_2(X), \dots, F_{n-1}(X)]^T=0, \quad (4.3)$$

where 0 is an n-1 dimensional vector $[0, 0, \dots, 0]^T$.

In general, with the existence of the solution of (4.3), by using Newton's iterative method, we can find the approximation of the solution X by the following iteration:

$$F(X_k) + F'(X_k)(X_{k+1} - X_k) = 0, \quad (4.4)$$

$$\begin{aligned} X_{k+1} - X_k &= -F'(X_k)^{-1} F(X_k) \\ &= -DF(X_k)^{-1} F(X_k), \quad k = 0, 1, 2, \dots \end{aligned}$$

where

$$DF(X) = \begin{bmatrix} M_1 & f(x_2) & 0 & & & \\ f(x_2) & M_2 & f(x_3) & & & \\ & f(x_3) & M_3 & f(x_4) & & \\ & & \ddots & \ddots & \ddots & \\ 0 & & & \ddots & \ddots & \\ & & & & 0 & f(x_{n-1}) & M_{n-1} \end{bmatrix} \quad (4.5)$$

and $M_i = -2f(x_i) + (x_{i-1} - x_i)f'(x_i)$ for $i = 1, 2, \dots, n-1$. $DF(X)$ is a symmetric tridiagonal matrix and Eq. (4.3) is equivalent to the linear system equations

$$DF(X_k)(X - X_k) = -F(X_k) \quad (4.6)$$

By using LU decomposition method [1], we can solve the linear equations (4.6) to obtain X_{k+1} . We know that the complexity of this method is $O(n)$, where n is the order of the tridiagonal matrix. We can solve (4.6) for sequence $\{X_k\}$, $k = 1, 2, \dots$. After N steps of iteration, the complexity is $N \cdot O(n) = O(n)$. That is, we can obtain the arbitrary approximation of the solution in polynomial time. We can use

$$\|X_{k+1} - X_k\|_2^2 = \sum_{i=1}^{n-2} (x_{k+1,i} - x_{k,i})^2 \leq \varepsilon, \quad \varepsilon > 0. \quad (4.7)$$

to find the number of steps N . Hence the result satisfies the precision requirement.

In Figure 5, we show the algorithm for finding the optimal window partition (from $x_0 = T(B_0)$ to $x_n = T(B_n)$) for any reconnection pattern $f(x)$.

Example 1. Assume that the reconnection pattern has a uniform distribution within the window, that is, $f(x) = c$, where c is a constant. From the above theorem, we have

$$g(\vec{X}) = \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} x_i f(x) dx = c \sum_{i=0}^{n-1} x_i (x_{i+1} - x_i)$$

$$F_i(\vec{X}) = \int_{x_i}^{x_{i+1}} c dx + c(x_{i-1} - x_i) = c(x_{i+1} - x_i + x_{i-1} - x_i) \quad \text{for } i = 1, 2, \dots, n-1.$$

If $F_i(\vec{X}) = 0$, we have $x_{i+1} - x_i = x_{i-1} - x_i$, then $x_i = x_0 + (i/n)$. This indicates evenly spaced intervals gives the optimal partition when the reconnect distribution is uniform within the window.

Example 2. Assume we are to divide a window of size 10 into 3 subintervals, and the reconnect distribution follows the formula

$$y = \begin{cases} x, & 0 \leq x \leq 5 \\ 10 - x, & 5 \leq x \leq 10; \end{cases}$$

Then, from the above theorem, we know that there exists a maximum value of $g(X)$, where $g(X) = x_1 \int_{x_1}^{x_2} y dx + x_2 \int_{x_2}^{10} y dx$, and $0 \leq x_1 < x_2 \leq 10$.

If $0 \leq x_1 \leq 5, 5 < x_2 \leq 10$, $g(X) = \frac{1}{2}x_2(x_2 - 10)^2 - \frac{1}{2}(x_2 - 10)^2 x_1 + 25x_1 - \frac{1}{2}x_1^3$.

If $0 \leq x_1 < x_2 \leq 5$, $g(X) = \frac{1}{2}x_1(x_2^2 - x_1^2) + \frac{1}{2}x_2(50 - x_2^2)$.

If $5 < x_1 < x_2 \leq 10$, $g(X) = \frac{1}{2}(x_1 - x_2)(x_2 - 10)^2 - \frac{1}{2}(x_2 - 10)^2 x_1$.

From the above equation, we obtained that $g(X)$ has the maximum value of 86.9385 when $x_1 = 3.1008$ and $x_2 = 5.4005$. Thus, we will divide the window at 3.1008 and 5.4005.

Algorithm: Optimal-Partition ($f(x), \varepsilon$)

Input: the reconnection distribution $f(x)$; the error limit ε ;

Output: the optimal partition of the interval: x_1, \dots, x_{n-1} ;

Step1. Evaluate $g(X) = \sum_{i=0}^{n-1} \left(\int_{x_i}^{x_{i+1}} x_i f(x) dx \right)$.

If the maximum of $g(X)$ at $x_0 \leq x_1 \leq \dots \leq x_{n-1} \leq x_n$ is found directly,
return x_1, \dots, x_{n-1} ;
otherwise, go to step 2.

Step 2. Solve nonlinear equation (4.2).
If we can find the exact solution, return x_1, \dots, x_{n-1} ;
otherwise; go to step 3 for an approximation solution.

Step 3. For a given approximation error ε , let $k=0$, and the initial value $X_0 = [x_{0,1}, x_{0,2}, \dots, x_{0,n-1}]^T$,
where $x_{0,i} = a + i * (b - a)/n$ for $i = 1, 2, \dots, n - 1$

Loop: Solve (4.6) for X_{k+1} by using LU decomposition

if $\|X_{k+1} - X_k\|_2^2 = \sum_{i=1}^{n-2} (x_{k+1,i} - x_{k,i})^2 > \varepsilon$, $k = k+1$, continue;

else return X_{k+1} .

Figure 5. Optimal Partition of Interval

4.3. A Dynamic Scheme

Like the dynamic BS scheme [16], we can modify our method of deriving partition of window period using reconnection pattern a little bit to further reduce the size of the report when

the number of items updated is small. That is, instead of using an N-bit sequence for B_0 , we explicitly use the IDs of items that have been update since $T(B_0)$. Other bit sequences (B_1, \dots, B_{n-1}) are composed as before. If both IDs and timestamps are implemented as 32-bit integers, the overall size of the report is $32k + \sum_{i=1}^{i=n-1} |B_i| + 32n$, where the first term is the size of k IDs (or B_0), the second term is the size of the rest of the bit-sequences (i.e., B_1, \dots, B_{n-1}), and the last term is the size of n timestamps. Note that the number of timestamps (or bit sequences) in our approach is, in general, different from that of Jing's. We will pick up this issue later.

5. Composing a Near-Optimal Invalidation Report

It has been proved that given a reconnection pattern, an invalidation report composed by the above method can minimize the false invalidation rate. However, the client reconnection pattern generally changes from one period to another or even from one window to another. To devise optimal invalidation reports, the server must continuously recalculate the partition for each window. This can cause tremendous overhead to the system, especially when the complexity of Newton's iterative method is unknown (because the complexity is dependent upon the reconnect distribution function $f(x)$); the only thing we know about the calculation is that each step of the iterative method has a polynomial time $O(n)$ complexity, where n is the order of the tridiagonal matrix (please refer to Equation (4.5) of Section 4.2). As a result, a near-optimal, but less costly and much faster solution to this problem may be desirable.

5.1 Approximation to the Reconnection Patterns

Clients' reconnection pattern, in mathematical functional form, is required when deriving the optimal partition using Newton's method. In reality, clients' reconnection distributions could be arbitrary. Finding well-fitting analytical formulas for arbitrary reconnect distributions is not a trivial job, not to mention the distributions change constantly. As a result, it is maybe desirable to describe a distribution as a series of numerical values. For example, the distribution in Figure 4 can be approximated by m sampling points: $(x_1, f(x_1)), (x_2, f(x_2)), \dots, (x_i, f(x_i)), \dots, (x_m, f(x_m))$. The larger the m value, the closer the approximation. Hereafter, we shall assume the reconnection pattern is a series of numerical values, which supposedly can be readily derived from the server's log.

5.2 Near-Optimal Partition of the Window

Recall that in the previous proof, the critical problem was to find a vector $X = [x_1, x_2, \dots, x_{n-1}]^T$ such that $g(X) = \sum_{i=0}^{n-1} \left(\int_{x_i}^{x_{i+1}} x_i f(x) dx \right)$ attains its maximum, where n, x_0 and x_n are three constants, $x_0 \leq \dots \leq x_i \leq x_{i+1} \leq \dots \leq x_n$.

As observed from Section 4, there are two steps to finding an approximation to the optimal solution X. They are: (1) initialize the intervals with some X^0 , and (2) quickly adjust the initial values to the optimal positions, even without knowledge of the analytical formula for the distribution function $f(x)$.

5.2.1 Finding the Initial Partition

Recalling from the proof of the theorem, the conditions for $g(X) = \sum_{i=0}^{n-1} (\int_{x_i}^{x_{i+1}} x_i f(x) dx)$ to attain its maximum are $\frac{\partial g(X)}{\partial x_i} = \int_{x_i}^{x_{i+1}} f(x) dx + f(x_i)(x_{i+1} - x_i) = 0, i = 1, 2, \dots, n - 1$. Let $S(x_{i-1}, x_i)$ be the area under the distribution curve bounded by subinterval $[x_{i-1}, x_i]$ on X-axial as shown in Figure 6.

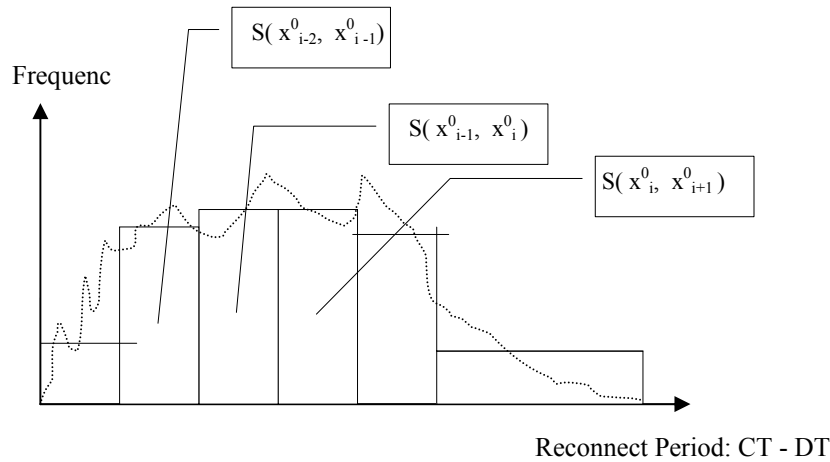


Figure 6: Finding Initial Value for X

Here, we assume the reconnect distribution is a series of numerical values. Let a and b be the start and end points on X-axial of the curve respectively. There exists a vector $X_0 = [a, x_1, x_2, \dots, x_{n-1}, b]^0$ such that these values divide evenly the whole area covered by the distribution curve into n subintervals. That is:

$$S(a, x_1^0) = \dots = S(x_{i-1}^0, x_i^0) = S(x_i^0, x_{i+1}^0) = \dots = S(x_{n-1}^0, b) = S(a, b) / n$$

As shown in the Figure 6, the dot line represents the true distribution curve. The rectangle areas are equal to their corresponding $S(x_i^0, x_{i+1}^0)$. The solid-line curve (called $f_0(x)$ thereafter) in Figure 6 can be regarded as an approximation of the actual distribution curve. Obviously, $f_0(x)$ also satisfies such equation:

$$\int_{x_i^0}^{x_{i+1}^0} f_0(x) dx + f_0(x_i^0)(x_{i+1}^0 - x_i^0) = 0, i = 1, 2, \dots, n - 1$$

because $\int_{x_i^0}^{x_{i+1}^0} f_0(x) dx = S(x_i^0, x_{i+1}^0) = S(x_{i-1}^0, x_i^0) = f_0(x_i^0)(x_i^0 - x_{i-1}^0)$. This demonstrates that vector

$X_0 = [a, x_1, x_2, \dots, x_{n-1}, b]^0$ is the ideal solution for $g_0(X) = \sum_{i=0}^{n-1} (\int_{x_i}^{x_{i+1}} x_i f_0(x) dx)$ to attain its maximum.

Here, we have to compromise on the condition of $f(x)$, that is, $f(x)$ should be continuous function, as required by the theorem. But the discussion here only serves the purpose of finding a good initial value; it will not affect the validity of the initial values for the partition or adjustment of these values later. Just as demonstrated in Figure 6, the process of finding the initial value of X should be straightforward.

5.2.2 Refining the Partition

Assume that the distribution curve $f(x)$ is comprised of a certain number, say m , of points, and the points are P_1, P_2, \dots, P_m with coordinates $(x_1, f(x_1)), (x_2, f(x_2)), \dots, (x_m, f(x_m))$. Because the interval $[x_i, x_{i+1}]$ is relatively tiny for a large m , according to trapezoidal rule in numerical analysis [1], we have the following approximation with relatively good precision:

Then the following values are easy to get:

$$\left. \begin{aligned} S(a, x_1) \\ S(a, x_2) = S(a, x_1) + S(x_1, x_2) \\ \dots \\ S(a, x_m) = S(a, x_{m-1}) + S(x_{m-1}, x_m) \\ S(a, b) = S(a, x_m) + S(x_m, b) \end{aligned} \right\} \quad (5.1)$$

because vector $X_0 = [a, x_1, x_2, \dots, x_{n-1}, b]^0$ divides evenly the whole area $[a, b]$ covered by the distribution curve into n subintervals. That is $S(a, x_1^0) = \dots = S(x_{i-1}^0, x_i^0) = \dots = S(x_{n-1}^0, b) = S(a, b) / n$. We should have

$$\left. \begin{aligned} S(a, x_1^0) &= S(a, b) / n \\ S(a, x_2^0) &= 2 \cdot S(a, b) / n \\ \dots \\ S(a, x_{n-1}^0) &= (n-1) \cdot S(a, b) / n \end{aligned} \right\} \quad (5.2)$$

With Equations (5.1) and (5.2), round x_i^0 to the nearest x_j , that is,

$$S(a, x_i^0) = i \cdot S(a, b) / n \approx S(a, x_j).$$

The value of $[a, x_1, x_2, \dots, x_{n-1}, b]^0$ is then obtained. The complexity of the above process to get X_0 is obviously $O(m)$.

Again, in order to obtain $\int_{x_i}^{x_{i+1}} f(x)dx$, we calculate $S(x_{i-1}, x_i)$ as described in Section 4.

Depending on the accuracy desired, we divide the whole period $[a, b]$ evenly into $r \times n$ subintervals, where r is an arbitrary positive integer that meets the condition $r \times n < m$, by letting $X_r = [a, x'_1, x'_2, \dots, x'_{r \times n-1}, b]^r$. Then, we have $S(x'_i - x'_{i-1}) = S(a, b) / (r \times n)$.

Next, adjust x_1^0 onto the x'_i values, where x'_i falls in (a, x_2^0) ; that is, adjust the value of x_1^0 once at a time from x'_1 to $x'_{2 \times r-1}$. After adjusting x_1^0 value to x'_i , calculate $f(x_1^0) \cdot (x_1^0 - a)$; a new value of x_2^0 is obtained according to $f(x_1^0) \cdot (x_1^0 - a) = S(x_1^0, x_2^0)$.

Calculating $f(x_2^0) \cdot (x_2^0 - x_1^0)$ using the updated x_2^0 value, a new value of x_3^0 is also obtained according to the formula $f(x_2^0) \cdot (x_2^0 - x_1^0) = S(x_3^0, x_2^0)$.

Repeat such procedure till the new x_{n-1}^0 is obtained. Calculate $g(X^0)$ according to the new $X^0 = [a, x_1, x_2, \dots, x_{n-1}, b]^0$. Find the corresponding X^0 so that $g(X^0)$ attains the maximum by comparing $g(X^0)$ value with different X^0 . To improve the accuracy, we can divide the two adjacent intervals in which the above x_1^0 is resided into finer intervals. After repeating the same process as above, higher accuracy can be achieved.

The complexity of above adjustment process is $O(r \times n \times m) = O(m^2)$, where r is the number of subintervals after the partition; n is a positive integer used for finer partition, which meets the condition $r \times n < m$; m is the total number of discrete values describing the distribution.

6. Simulation Results

In this section, we report simulation results on Jing's and our approaches based on the size of invalidation reports and false invalidation rate. We have chosen to experiment with the dynamic versions of these two approaches because of their flexibility in accommodating a variable number of updates in the report. The results should be applicable to the original schemes without much difference. A report has two parts – bit-sequences and timestamps. Since item IDs are used as the first bit-sequence B_0 in the dynamic schemes of both approaches, we shall exclude B_0 from the “bit-sequences” in the following discussion, unless otherwise stated. We will also discuss the effect of timestamps on the overall size of a report later.

The purpose of the simulations is mainly to compare the size of the reports and the effectiveness of the bit-sequences in reducing the false invalidation rate, denoted FIR, which is defined as
$$FIR = \frac{\text{number-of-items-falsely-invalidated}}{\text{number-of-items-invalidated}}.$$

Since the compositions of the bit-sequences are different in these two approaches, the lengths of bit-sequences will also be different. In order to make fair comparisons of the effectiveness of bit-sequences in correctly invalidating items, we define an effectiveness measure, denoted EF, as
$$EF = \frac{1 - FIR}{\text{length-of-bit-sequences}}.$$

This measure tells the percentage of correct invalidations per unit length of bit-sequences. We shall compute the relative effectiveness (REF), the ratio of our EF to Jing's EF, as
$$\frac{\text{ours-}EF}{\text{Jing's-}EF},$$
 to show the how effective our approach is when compared to Jing's approach.

It is noted that the near-optimal method proposed does not require any a priori knowledge of the reconnection pattern. Approximate clients' reconnect distribution is derived dynamically from clients' reconnect activities. In the simulation, we draw 6,000 sampling points to approximate the reconnect distribution. To obtain the optimal partition, we first divide the window into 3 intervals, as described in Section 5, and then divide each interval into 10 subintervals to obtain a more accurate partition; finally, we divide the two candidate subintervals that contain the optimal partition points, into 10 intervals each to derive our final partition. The entire process of deriving a near-optimal partition takes less than 0.01 ms on a PC with Celeron 400 MHZ processor and 128 MB RAM. The flexibility and speed of our method justify its application to real mobile database environments.

For ease of understanding, we have chosen the two patterns, a uniform distribution and a non-uniform distribution with a peak in the middle of the window described in the Examples 1 and 2 of Section 4.1, for our simulations. Certainly, our near-optimal partition can apply to any distribution function. It is noted that cache size has no effect on FIR, which is due to the inaccuracy of the report. We have also tested with various database update rates, 10%, 30%, 50%, which are the percentages of items updated during the last window period, to see their effects on the false invalidation rate.

The lengths of the bit-sequences in the two approaches are usually different. As mentioned earlier, the expected size of Jing's bit-sequences (excluding B_0) can be calculated beforehand as $2k$, where k is the number of items updated since $T(B_0)$; in our approach, it depends on the number of subintervals in the window, which can be chosen as desired. In order to compare the effectiveness of bit-sequences, we have chosen the number of subintervals to be 3 in our approach so that the lengths of our bit-sequences can be as close to Jing's as possible.

Note that in general, the more subintervals in a window, the larger the reports, and the fewer the false invalidations. In our method, we can divide the window into more subintervals as desired to reduce the false invalidation rate, while Jing’s cannot (size = 2k).

Table 1: Uniform Distribution

	10%		30%		50%	
	Length	FIR	Length	FIR	Length	FIR
Near-Opt	1731	0.1883	5065	0.1884	8397	0.1881
Optimal	1731	0.1883	5065	0.1884	8397	0.1880
Jing’s	2281	0.2008	6344	0.2001	10378	0.2005
Ratio	0.7589	0.9379	0.7984	0.9412	0.8091	0.9378
REF	1.337		1.271		1.253	

As discussed earlier in Section 4, for a uniform reconnection pattern, an evenly divided window yields the optimal performance. For the convenience of calculation, the window size has been set to be 10 time units in all simulations. In the following tables, in the “Length” column we report the length of bit-sequences in bits. The row “Ratio” shows the ratios of the length and FIR of the optimal method to Jing’s. Since the performance of our near-optimal solution is nearly identical to the optimal method, the ratios also apply to both the optimal and near-optimal methods to Jing’s. Hereafter, we shall not differentiate our near-optimal and the optimal methods, unless otherwise stated.

It can be observed from Table 1 that our bit-sequence size is around 80% of Jing’s (i.e., 0.7589, 0.7984, 0.8091 for 10%, 30%, 50% update rates, respectively). This result is consistent with our analysis on the estimations of bit-sequence sizes. Recall that the length of Jing’s bit-sequences is $2k$ (excluding B_0), while ours is $k + (2/3)k = (5/3)k$, where k is the size of B_1 (and also the number of items updated), and $(2/3)k$ is the expected size for B_2 . That is, ours is only 83% ($(5/3)k / 2k \approx 0.83$) of Jing’s.

The FIRs basically remain the same for different database update rates in each approach, that is, around 18.8% in our approach and 20.0% in Jing’s approach. It indicates that FIR has to do with the ways of composing bit-sequences, and has nothing to do with the rates of updates.

In summary, not only are our bit-sequences shorter than Jing’s, approximately 80% of Jing’s, but also achieve better (or lower) false invalidation rates (approximately 94% of Jing’s). This implies our bit-sequences are more effective in lowering the false invalidation rate. According to the relative effectiveness measure REF, our bit-sequences are 25 – 34% more effective in correctly invalidating items than Jing’s.

Now let us consider the size of timestamps. The total size of timestamps in Jing’s report is $32\log(k)$, while it is $32n$ in ours, where $\log(k)$ and n are the numbers of bit-sequences in respective reports. In our report, there are 3 (i.e., $n = 3$) timestamps, while in Jing’s report, it has $\log(k)$ timestamps ($\log(1,000)=10, \dots, \log(5,000)=13$ for 1,000, ..., 5,000 updates, respectively, during the last window). Clearly, we use much less timestamps and consume less space than Jing’s report.

In Table 2, we show the results of the non-uniform distribution described in Example 2 of Section 4.1. According to Theorem 1, we divided the window at 3.1008 and 5.4008. Again, our bit-sequences are shorter (about 80% of Jing’s), and yet achieve much lower false invalidation rates (76% of Jing’s). As in the uniform case, the FIRs remain the same in each approach for

different item update rates. The FIRs are around 17.6% in our approach, compared to 23.1% in Jing's. In terms of the REF measure, our bit-sequences are 31- 40% more effective than Jing's.

Table 2: Non-uniform Distribution

	10%		30%		50%	
	Length	FIR	Length	FIR	Length	FIR
Near-Opt	1754	0.1757	5134	0.1757	8513	0.1759
Optimal	1754	0.1755	5134	0.1755	8513	0.1757
Jing's	2281	0.2315	6344	0.2310	10378	0.2309
Ratio	0.7690	0.7581	0.8093	0.7598	0.8203	0.7610
REF	1.395		1.325		1.307	

It is worth mentioning that if there is enough bandwidth for longer reports, in our approach we can easily divide the window into more subintervals (i.e., more bit-sequences) to achieve lower false invalidation rates. However, this may not be possible for Jing's approach because the number of bit-sequences (i.e., $\log(k)$) is completely determined by the number of updates (k) in the window period. That is, even though there is still bandwidth left for use, Jing's approach simply cannot use it to reduce the false invalidation rates. Excess bandwidth may be used to reduce false invalidation by increasing the number of bit-sequences in our approach.

In summary, our approach clearly outperforms Jing's approach in terms of the length of bit-sequences, number of timestamps, effectiveness of reducing FIR, and flexibility in using excess bandwidth to reduce false invalidation rates. Our near-optimal solution yields almost identical results as the optimal one. As mentioned earlier, the near-optimal solution can be applied to any distributions without a priori knowledge of the distributions; it is fast and agile.

7. Conclusions

In this paper, we discussed the composition of invalidation reports to achieve minimal false invalidation rates. We have found that false invalidation rates have to do with the reconnection patterns of clients, that is, the distribution of the durations between disconnections and reconnections. Based on the bit-sequence approach [16], we redesigned the hierarchy of the bit-sequences, taking into account the reconnection patterns of mobile clients. We have shown that by using Newton's method, we can derive a partition of the report window to achieve a minimal false invalidation rate for a given reconnection pattern. In order for this approach to be used practically in the mobile database environment, we devise a near-optimal partition of the window, which does not require a priori knowledge of the reconnection pattern. It is dynamic, fast, and yields almost identical results as the optimal one. We have performed simulations to show that our approach of composing reports indeed has better performance than Jing's in terms of the length of bit-sequences, number of timestamps, effectiveness of reducing FIR, and flexibility in using excess bandwidth.

Acknowledgement

Thanks go to Meng Su for his earlier contributions to the paper.

References

1. Axelsson, "Iterative Solution Methods", *Cambridge University Press*, 1994.

2. D. Barbara, "Mobile Computing and Databases-A Survey", *IEEE Transactions on Knowledge and Data Engineering*, pp. 108-117, Vol. 11, No. 1, Jan/Feb, 1999
3. D. Barbara, T. Imielinski, "Sleepers and workaholics: Caching strategies for mobile environments", *Pro. ACM SIGMOD Conf, on Management of Data*, pp.1-12, May, 1994.
4. C. Bornhövd, M. Altinel, S. Krishnamurthy, C. Mohan, H. Pirahesh, B. Reinwald, "DBCACHE: Middle-tier Database Caching for Highly Scalable E-business Architectures", *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*.
5. Budiarto, S. Nishio, M. Tsukamoto, "Data Management Issues in Mobile and Peer-to-peer Environments", *Data and Knowledge Engineering*, v.41 n.2-3, pp.183-204, 2002.
6. J. Cai, K. Tan, B. Ooi, "On Incremental Cache Coherency Schemes in Mobile Computing Environments", *Proc. of IEEE Data Engineering*, Pg. 114-123, April, 1997
7. M. J. Carey, M. J. Franklin, M. Livny & E. J. Shekita, "Data Caching Tradeoffs in Client-Server DBMS Architectures", *Proc. of ACM 1991 SIGMOD*, pp. 357-366, May, 1991.
8. H. Chung, H. Cho, "Data Caching with Incremental Update Propagation in Mobile Computing Environments", *Proc. Australian Workshop on Mobile Computing and Databases and Applications*, pp. 120-134, Feb. 1996.
9. A. Elmargamid, J. Jing, and T. Furukawa, "Wireless Client-Server Computing for Personal Information Services and Applications", *ACM SIGMOD Record*, pp. 43-49, Dec. 1995.
10. M. Franklin, M. Carey, M. Livny, "Global Memory Management in Client-Server DBMS Architectures". *Proc. of VLDD*, pp. 596-609, August 1992.
11. C. G. Gray, D. R. Cheriton, "Leases: An Efficient Fault-Tolerant Mechanism for Distributed File Cache Consistency". *Proc. of SOSR*, pp. 202-210, Feb. 1989.
12. J. Holliday, D. Agrawal, A. El Abbadi, "Planned Disconnections for Mobile Databases", *Proceedings of 11th International Workshop on Database and Expert Systems Applications*, pp. 165 – 169, 2000.
13. W-C. Hou, C. Wang, M. Su, "Composing Optimal Invalidation Reports for Mobile Databases", *Journal of Distributed Information Management*, Vol. 3, No. 2, June 2005, pp. 126 – 132.
14. Q. Hu, D. Lee, "Cache Algorithm based on Adaptive Invalidation Reports for Mobile Environments", *Cluster Computing*, Vol. 1. No. 1, pp. 39 -48, Feb. 1998.
15. T. Imielinski, S. Vishwanath, B. R. Badrinath, "Energy efficient indexing on air", *Proceedings of the ACM SIGMOD Conference on Management of Data*, Minneapolis, Minnesota, 1994.
16. J. Jing, A. K. Elmagamid, A. Helal, R. Alonso, "Bit-Sequences: An Adaptive Cache Invalidation Method in Mobile Client/Server Environments". *ACM/Baltzer Journal of Mobile Network and Applications*, 2(2), pp.115-127, 1997.
17. A. Kahol, S. Khurana, S. K. S. Gupta, P. K. Srimani, "A Strategy to Manage Cache Consistency in a Disconnected Distributed Environment." *IEEE Trans.Parallel Distrib.Syst.* 12(7):686-700, 2000.
18. K. Y. Lai, Z. Tari, P. Bertok, "Cost Efficient Broadcast Based Cache Invalidation for Mobile Environments", *18th ACM Symposium on Applied Computing*, pp. 871-877, 2003.
19. K.C.K. Lee, V. L. Hong, A. Si, "Semantic Data Broadcast for a Mobile Environment Based on Dynamic and Adaptive Chunking", *IEEE Transactions on Computers*, Vol. 51, No. 10, pp. 1253 – 1268, 2002.

20. Y. S. Lu, X. K. Shao, "Improve Performance of Disconnected Operation through Submitting by Probability and Transferring Transactions in Groups", *International conference on Computer Networks and mobile Computing*, pp. 502-505, 2003.
21. P. Nuggehalli, V. Srinivasan, C.-F. Chiasserini, "Energy-efficient Caching Strategies in Ad Hoc Wireless Networks", *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pp. 25-34, 2003.
22. X.-K. Shao, Y.-S. Lu, "Maintain Cache Consistency in Mobile Database Using Dynamical Periodical Broadcasting Strategy", *Proc of 2nd International Conference on Machine Learning and Cybernetics*, pp. 2389-2393, 2003.
23. M. Stonebraker, et al, "Third -Generation Data Base System Manifesto," *SIGMOD Record* 19, 3, pp. 241- 234, Sept. 1990.
24. J. Strain, R. Acuff, T. Rindfleisch, L. Fagan, "A Pen-Driven, Mobile Surgical Database: Design and Implementations," <http://www-smi.stanford.edu/projects/mobile/amia94-2.html>, 1994.
25. A. Waluyo, B. Srinivasan, D. Taniar, "A Taxonomy of Broadcast Indexing Schemes for Multi Channel Data Dissemination in Mobile Databases", *18th International Conference on Advanced Information Networking and Applications (AINA)*, Vol. 1, pp. 213 – 218, 2004.
26. Y. Wang, "Cache Consistency and Concurrency Control in a Client/Server DBMS Architecture", *Proc. of ACM SIGMOD 1991*, pp. 367-376, May, 1991.
27. Z. Wang, S. Das, H. Che, M. Kumar, "SACCS: Scalable Asynchronous Cache Consistency Scheme for Mobile Environments", *Proc of 23rd International Conference on Distributed Systems Workshop*, pp. 797-802, 2003.
28. K. Wilkinson, M. Neimat, "Maintaining Consistency of Client-Cached Data", *Proc. of 16th VLDB*, pp. 122-133, Aug.1990.
29. K. Wu, P. Yu, M. Chen, "Energy-efficient Caching for Wireless Mobile Computing", *Proc. 12th International Conference on Data Engineering*, pp. 34-50, Feb. 1996.



Wen-Chi Hou received the MS and PhD degrees in computer science and engineering from Case Western Reserve University, Cleveland Ohio, in 1985 and 1989, respectively. He is presently an associated professor of computer science at Southern Illinois University at Carbondale. His interests include statistical databases, mobile databases, XML databases, and data streams.



Chih-Fang Wang received his PhD in computer engineer at University of Florida in 1998. His current research interests include sequential, parallel and distributed algorithms, data structures, high performance computing, optical networks, quantum computing, DNA computing, bioinformatics, wireless/mobile security, data mining, and mobile agents and secure mobile agent platforms.

Haiqing Li was a Ph.D student in the department of mathematics and M.S student in the computer science department at Southern Illinois University, Carbondale, IL.