An Approach of Composing Near Optimal Invalidation Reports

Meng Su
Department of Computer Science Penn State Erie
The Behrend College
Erie, PA 16509, USA
mengsu@psu.edu

Chih-Fang Wang
Department of Computer Science
Southern Illinois University
Carbondale, IL 62901, USA
cfw@cs.siu.edu

Wen-Chi Hou
Department of Computer Science
Southern Illinois University
Carbondale, IL 62901, USA
hou@cs.siu.edu

ABSTRACT
Caching can reduce expensive data transfers and improve the performance of mobile computing. In order to reuse caches after short disconnections, invalidation reports are broadcast to clients to identify outdated items.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 may set in if the accurate timing of updates is not provided. In this research, we aim to reduce the false invalidation rates of cached items. Based on our analysis, false invalidation rates are closely related to clients' reconnection patterns (i.e., the distribution of the time spans between disconnections and reconnections). We show that in theory for any given reconnection pattern, a report with a minimal false invalidation rate can be derived. For practical uses, we propose to capture the reconnection pattern by sampling and develop a method to compose a near-optimal invalidation report. This method is simple and fast. Experimental results have confirmed that our method is indeed more effective in reducing the false invalidation rate than others.

Categories and Subject Descriptors
H.1.1 [**Systems and Information Theory**]: *Information theory, Value of information.*

General Terms
Algorithms, Design, Theory.

Keywords
Mobile Databases, Invalidation Report, False Invalidation, Reconnection Pattern.

MDM 2005 05 Ayia Napa Cyprus

1. INTRODUCTION
It was predicted [3] that in a mobile wireless computing environment of the future, massive numbers of low powered palmtop machines would query databases over the wireless communication channels. Technology progresses so fast that it is now the reality. Numerous such mobile databases are in use today, including mobile surgical databases [23], stock information systems, and a variety of general purposed pocket databases for small business and personal use. In a mobile computing environment [5], a set of database servers disseminates information via wireless channels to mobile clients [24]. 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).

It is generally assumed that updates to database are broadcast without much delay to the clients (by the server). Therefore, as long as the clients stay connected, their caches are current. However, in a mobile database environment, clients are frequently disconnected due to some battery power saving measures [12] or unpredictable failures. Discarding entire caches after short disconnections can 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 schemes which helps in power saving and also for the reuse of cache is much needed [20].

Many caching coherence algorithms have been proposed for conventional client-server architectures [7, 10, 11, 25, 27]. There have been some researches on cache management for mobile computing published recently in the literature [2, 6, 8, 9, 17, 18, 19, 21, 28, 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, 15]. 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 continue the discussion of

how to compose a report with a minimal false invalidation rate [13]. We have observed that false invalidation rates have to do with clients' reconnection patterns (i.e., the distribution of the time spans between disconnection and reconnection). By applying nonlinear system's approximation method to the clients' reconnection pattern, we showed 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 computed efficiently. Simulation results have also confirmed that our near-optimal method is fast and effective in reducing false invalidations.

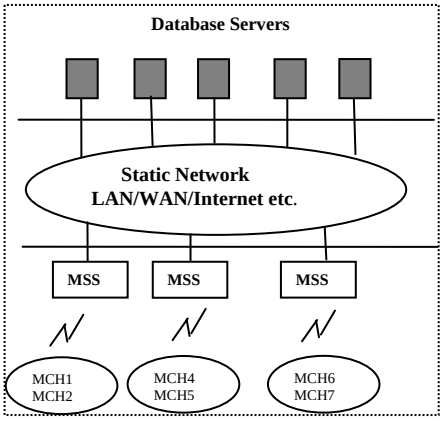
The rest of the paper is organized as follows. In Section 2, we review the mobile computing environment and research on cache management for such an environment. Section 3 reviews the compositions of invalidation reports to lay down the groundwork for our approach. In Section 4, we propose the algorithm to take clients' reconnection patterns into account in the design of invalidation reports. Unfortunately, deriving such optimal reports requires a priori knowledge of the reconnection patterns and much computation. Therefore, we propose to compose a near-optimal report in Section 5. The solution is practical, dynamic, fast, and agile. Simulation results are presented in Section 6, and we conclude our study in Section 7.

2. PRELIMINARY

2.1 A Mobile Computing Architecture

With networks of powerful workstations becoming commonplace, client-server software architectures efficiently provide access to shared services and resources [22]. In a client-server architecture, the database resides at the server and is accessed by application programs running on the clients' workstations. A client communicates with the database server through message passing. It, however, increases the cost of each data request. A solution is to reduce the number of requests by caching a portion of the database at the client side.

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 [14]. It consists of two distinct sets of entities: a large number of mobile client hosts (MCHs) and relatively fewer but more powerful fixed hosts, which are connected through a wired network. MCHs usually run on small batteries, such as AA, while fixed hosts may have continuous power supplies. 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 mobile client hosts, which are located within a radio coverage area, called a wireless cell. A cell could be a real cell as in a cellular communication network or a wireless local area network, which operates within the area of a building. In the former case, the bandwidth could be severely limited, supporting a data rate in the range of 10 to 20 Kb/sec, while in the latter, the bandwidth would be much higher, up to 10Mb/sec [1]. 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. At any time, an MCH may logically belong to only one cell. 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.



MSS: Mobile Support Station, MCH: Mobile Client Host Note: this figure is modified from [3].

Figure 1: A Mobile Computing Architecture

2.2. Cache Management

Caching on the client side can reduce client-server interactions, lessening the network traffic and message-processing overhead for both the servers and clients. Various cache coherence schemes [5, 9, 15, 16, etc.] have been developed for conventional client-server architectures. These algorithms are generally classified into two categories: the callback approach and the detection approach. In the callback approach, servers are responsible for the coherency of caches. Usually, servers send invalidation messages directly to clients to invalidate data items cached. On the other hand, in the detection approach, it is the clients' responsibility to maintain the consistency.

Clients request servers, from time to time, to validate their cached data. Since mobile client hosts frequently disconnect to conserve battery power and are frequently on the move, it is very difficult, for a server to keep track of the status and locations of the clients and the validity of cached data items. As a result, the callback approach cannot be easily implemented mobile environment. On the other hand, due to the limited transmission capability of mobile clients, it is not appropriate for clients to request information so frequently. Moreover, the narrow bandwidth [5] of the wireless network could be clogged up if a massive number of clients attempt to query the server to validate their caches. As a result, both the callback and detection approaches employed in the traditional client-server architecture are not readily applicable to the mobile environment, and new methods have to be designed.

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 [15, 17, 21, 26, etc].

Based on the timing of the invalidation messages being broadcast by the servers, cache invalidation methods can be either asynchronous or synchronous. In an asynchronous approach [16, 24, 26], a server broadcasts an invalidation message as soon as an item changes its value. A client who is connected can immediately invalidate these items in its cache. The biggest problem with the asynchronous approach is its unpredictable waiting time for such invalidation messages, which depends on the update activities on the database. Some improvements have been made in [20] such that even when the updating frequency is low, it is assured that at least one broadcast is sent in a certain period.

In the synchronous approach [3, 24, 27], 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. Some latency could be induced between the actual updates and notification of the updates to the mobile clients. Once invalid items are found in the cache, the client submits an uplink request for updated values. The invalidation report divides the time into intervals.

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.

3. INVALIDATION REPORTS FOR RECONNECTED USERS

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 entirely 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, 20]. In this research, we will focus on composing invalidation reports for reconnected clients. In the following, we briefly describe existing ways for composing invalidation reports.

3.1 Broadcasting Timestamp (BT) Strategy

It is generally assumed that a reconnected client cannot answer any queries until it has received the newest invalidation report and updated its cache.

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 of a report, the larger the invalidation report, which can lead to a long latency in the dissemination of reports. Normally, for each item cached, the client either purges it from the cache if the item is reported to have been changed, or changes the item's timestamp to the timestamp of the report (i.e., up to date). Note that the entire cache will have to be discarded if the client has disconnected longer than the period covered by the report.

3.2 Bit-Sequence (BS) Approach

In the above approach, updated items are indicated by IDs and their respective timestamps in the report. When the number of items updated is large, the size of the invalidation report can become large too. In order to save bandwidth, the bit-sequence (BS) approach is proposed [15]. Two techniques, the bit-sequence mapping and the update aggregation, are used to reduce the size of the report. The bit-sequence mapping aims to reduce the naming space of the items, while the update aggregation aims to reduce the number of timestamps used in the report. Since our approach is based on the BS approach, we shall elaborate on this approach a little more here.

In the BS approach, 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. This technique reduces the naming space from $Nlog(N)$ bits for N items (needed in BT approach) to N bits here.

Another technique, called the update aggregation [15], is used in conjunction with the bit-sequence mapping to reduce the number of timestamps in a report. Instead of having one timestamp for each updated item, the report uses only one timestamp, which is the time when the report is to be delivered. 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.

Although update aggregation reduces the size of a report, it 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 that are updated after d in the report, all the items mentioned in the report will all have to be purged when the client reconnects.

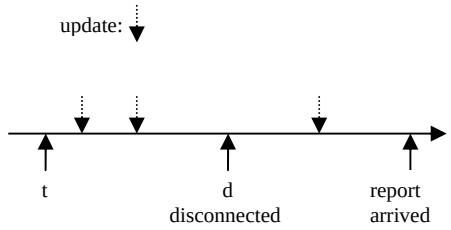


Figure 2: False Invalidations

In order to reduce false invalidations, a hierarchically structured and more detailed report has been proposed [15]. 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 ($0 \le i \le n-1$), 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 4, 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 been updated since $T(B_0)$ (i.e., a "1" bits in B_0). Again, half of the bits in B_1 (i.e., N/4 bits) have the value "1", indicating half of the N/2 items that have been updated since $T(B_0)$ were actually updated after $T(B_1)$. 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. It can be observed that the total number of bit-sequences n is log(N).

The issue of false invalidation doesn't exist in the architecture [16] because the servers can directly invalidate the client caches or clients can query the Mobile Support Station for the validation of caches after reconnection.

The above BS scheme has been modified in [11] 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, … B_{n-1}) are composed in the same way as in the original BS. That is, sequence B_i ($0 < i \le n-1$) has $k/2^{i-1}$ bits, with half of them being "1"s, where k is the number of items updated since $T(B_0)$.

4. COMPOSITION OF OPTIMAL HIERARCHICAL BIT-SEQUENCES

Although Jing's [15] 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. In fact, this "half-update-partition" scheme could favor some situations over others. 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 3, 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 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 our research [13], we considered 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 [19] of a client can be represented by a certain probability distribution f(x) such as the one shown in Figure 3. We analyzed 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., $T(B_i) \le x$). 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.

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

$$TFI = c\sum_{i=0}^{n-1}\left(\int_{T(B_i)}^{T(B_{i+1})}(x - T(B_i))\cdot f(x)\,dx\right) \qquad (4.1)$$

where n is the number of bit-sequences in the report, $T(B_n)$= CT (i.e., the current time). To illustrate our algorithm, we rewrite the theorem in [13] as following:

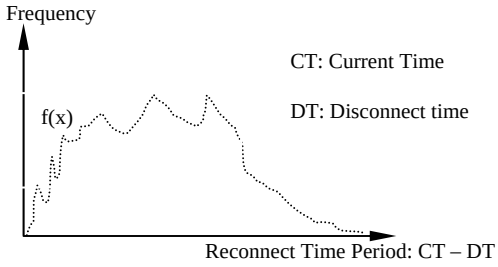


Figure 3: Reconnect Time Distribution

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 $X = [x_1, \cdots, x_{n-1}]^T$ such that

$g(X) = \sum_{i=0}^{n-1} (\int_{x_i}^{x_{i+1}} x_i f(x)dx)$ Obtains its maximum,

where $a = x_0 \le x_1 \le \cdots \le x_{n-1} \le x_n = b$. Furthermore, the value X is the solution of nonlinear system

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

In [13], we deduced that to minimize TFI is equivalent to maximize $\sum_{i=0}^{n-1} (\int_{T(B_i)}^{T(B_{i+1})} T(B_i) f(x)dx)$ .Hence, $T(B_i)$, i=1,… 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

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

where 0 is a n-1 dimension vector [0, 0, …, 0]^T.

In general, with the existence result 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:

$$F(X_k) + F'(X_k)(X_{k+1} - X_k) = 0,$$
$$X_{k+1} - X_k = -F'(X_k)^{-1} F(X_k)$$
$$= -DF(X_k)^{-1} F(X_k),\ \ k = 0, 1, 2, \cdots,$$

where

$$DF(X) = \begin{bmatrix} M_1 & f(x_2) & 0 & & & \\ f(x_2) & M_2 & f(x_3) & & 0 & \\ & f(x_3) & M_3 & f(x_4) & & \\ & & \ddots & \ddots & \ddots & \\ 0 & & & \ddots & \ddots & \ddots \\ & & & 0 & f(x_{n-1}) & M_{n-1} \end{bmatrix} \quad (4.4)$$

and

$M_i = -2 f(x_i) + (x_{i-1} - x_i) f'(x_i)$ for $i = 1, 2, \cdots, n-1$,

$DF(X)$ is a symmetric tridiagonal matrix and Eq. (4.3) is equivalent to the linear system equations

$$DF(X_k)(X - X_k) = -F(X_k) \quad (4.5)$$

By using LU decomposition method [1], we can solve the linear equations (4.5) 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 $\{X_k\}$, $k = 1, 2, \cdots$. 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

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

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

Here is the algorithm of finding the optimal window partition (from $x_0 = T(B_0)$ to $x_n = T(B_n)$) for any reconnection pattern f(x) based on the above analysis:

Algorithm:

Step1. Evaluate $g(X) = \sum_{i=0}^{n-1} (\int_{x_i}^{x_{i+1}} x_i f(x)dx)$. If we can find the maximum of g(X) at $x_0 \le x_1 \le \cdots \le x_{n-1} \le x_n$ directly, return $x_1, \ldots, x_{n-1}$. Otherwise go to step 2.

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

Step 3. For a given approximation error ε, let k =0, and the initial value

$X_0 = [x_{0,1}, x_{0,2}, \cdots, x_{0,n-1}]^T$,

where $x_{0,i} = a + i*(b - a)/n$ for $i = 1, 2, \cdots, n-1$

Loop:

Solve (4.5) 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} .

Example. Assume that we are to divide a window of size 10 into 2 subintervals, and the reconnect pattern follow the normal distribution

$$f(x) = \exp(-(x-5)^2/200), \quad 0 \le x \le 10.$$

$$g(X) = \int_{x_1}^{10} x_1 f(x)dx$$

The optimal partition point x_1 is the solution of equation (4.2) which is

$$\int_{x_1}^{10} f(x)dx - x_1 f(x_1) = 0$$

Let $\varepsilon = 10^{-5}$ and the initial value $x_{0,1} = 5$, by using the iteration of step 3 in the algorithm, we obtained $x_1 = 4.8997$.

5. A Practical Approach to Compose 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 [13]. However, the approximate solution by Newton iteration depends on the selection of the initial values. We can not guarantee that the initialization in step 3 always works. As a result, a new approximate algorithm for equation (4.2) which can replace step 3 is described in the following.

In reality, clients' reconnection distributions could be arbitrary. The real distribution is actually a series of numerical values. For example, the distribution in Figure 5 can be approximated by m sampling points: (x_1, y_1), (x_2, y_2), …, (x_i, y_i), … (x_m, y_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.

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

Let $S(x_{i-1}, x_i)$ denotes the area under the distribution curve bounded by subinterval $[x_{i-1}, x_i]$ on X-axial as shown in Figure 6. 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:

$$S(a, x^0_1) = \dots = S(x^0_{i-1}, x^0_i) = S(x^0_i, x^0_{i+1}) = \dots = S(x^0_{n-1}, b) = S(a, b)/n$$

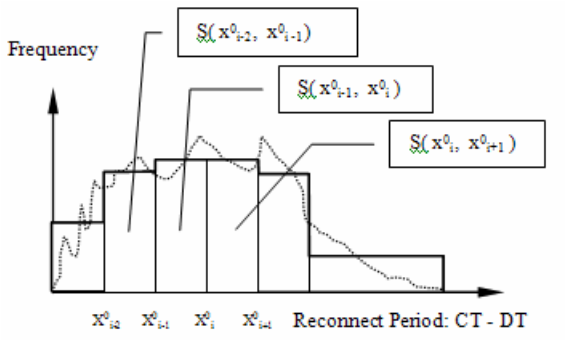


Figure 4: Finding Initial Value for X

As shown in the Figure 4, the dot line represents the true distribution curve. The rectangle areas are equal to their corresponding $S(x^0_i, x^0_{i+1})$. 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:

$$\int_{x^0_i}^{x^0_{i+1}} f_0(x)dx + f_0(x^0_i)(x^0_{i-1} - x^0_i) = 0, \; i = 1, 2, \cdots, n-1$$

because

$$\int_{x^0_i}^{x^0_{i+1}} f_0(x)dx = S(x^0_i, x^0_{i+1}) = S(x^0_{i-1}, x^0_i) = f_0(x^0_i)(x^0_i - x^0_{i-1})$$

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 obtain its maximum. Just as demonstrated in Figure 6, the process of finding the initial value of X should be straightforward.

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, ..., P_m with coordinates $(x_1, f(x_1))$, $(x_2, f(x_2))$, ..., $(x_m, f(x_m))$. 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:

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

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 $S(a, x^0_1) = \dots = S(x^0_{i-1}, x^0_i) = \dots = S(x^0_{n-1}, b) = S(a, b)/n$.

We should have

$$\left.\begin{array}{l} S(a, x^0_1) = S(a, b) / n \\ S(a, x^0_2) = 2 \cdot S(a, b) / n \\ \ldots \\ S(a, x^0_{n-1}) = (n-1) \cdot S(a, b) / n \end{array}\right\} \quad (5.2)$$

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

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

The value of $[a, x_1, x_2, \ldots, x_{n-1}, b]^0$ is then obtained.

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×n subintervals, where r is an arbitrary positive integer that meets the condition r×n < m, by letting $X_r = [a, x'_1, x'_2, \ldots, x'_{r\times n-1}, b]^r$. Then, we have

$$S(x'_i - x'_{i-1}) = S(a, b) / (r\times n),$$

Next, adjust x^0_1 onto the x'_i values, where x'_i falls in (a, x^0_2); that is, adjust the value of x^0_1 once at a time from x'_1 to $x'_{2\times r-1}$. After adjusting x^0_1 value to x'_i, calculate $f(x^0_1)(x^0_1 - a)$; a new value of x^0_2 is obtained according to

$$f(x^0_1) \cdot (x^0_1 - a) = S(x^0_1, x^0_2).$$

Calculating $f(x^0_2)(x^0_2 - x^0_1)$ using the updated x^0_2 value, a new value of x^0_3 is also obtained according to the formula

$$f(x^0_2) \cdot (x^0_2 - x^0_1) = S(x^0_3, x^0_2)$$

Repeat such procedure till the new x^0_{n-1} is obtained. Calculate $g(X^0)$ according to the new $X^0 = [a, x_1, x_2, \ldots, x_{n-1}, b]^0$. 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.

6. SIMULATION COMPARISON

In this section, simulation results on Jing's and our approaches based on the size of invalidation reports and false invalidation rate are compared. 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. We also discussed 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

$$\text{FIR} = \frac{number-of-items-falsely-invalidated}{number-of-items-invalidated}$$

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

$$\text{EF} = \frac{1-FIR}{length-of-bit-sequences}$$

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 $\frac{our-EF}{Jing's-EF}$, to show the how effective our approach is when compared to Jing's approach.

It is noted that the near-optimal method proposed in section 5 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, the database size is set to be 10,000. Furthermore, 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.

We chose 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 in [13] for our simulations. We have also tested with various database update rates, 10%, 20%, 30%, 40%, and 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. 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

Table 1: Uniform Distribution

Method	Update Rate	10%		20%		30%		40%		50%	
		Length	FIR	Length	FIR	Length	FIR	Length	FIR	Length	FIR
Near-Opt		1754	0.1757	3444	0.1760	5134	0.1757	6826	0.1755	8513	0.1759
Optimal		1754	0.1755	3444	0.1759	5134	0.1755	6826	0.1754	8513	0.1757
Jing's		2281	0.2315	4313	0.2307	6344	0.2310	8345	0.2313	10378	0.2309
Ratio		0.7690	0.7581	0.7985	0.7626	0.8093	0.7598	0.8180	0.7585	0.8203	0.7610
REF		1.395		1.341		1.325		1.311		1.307	

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).

Clearly, we use much less timestamps and consume less space than Jing's report. In Table 2, we show the results for the non-uniform distribution described in [13]. According to the Theorem, we divided the window at 3.1008 and 5.4008. Again,

Table 2: Non-uniform Distribution

Method	Update Rate	10%		20%		30%		40%		50%	
		Length	FIR	Length	FIR	Length	FIR	Length	FIR	Length	FIR
Near-Opt		1731	0.1883	3399	0.1879	5065	0.1884	6733	0.1882	8397	0.1881
Optimal		1731	0.1883	3398	0.1878	5065	0.1884	6732	0.1881	8397	0.1880
Jing's		2281	0.2008	4313	0.2004	6344	0.2001	8345	0.2006	10378	0.2005
Ratio		0.7589	0.9379	0.7879	0.9371	0.7984	0.9412	0.8067	0.9377	0.8091	0.9378
REF		1.337		1.289		1.271		1.259		1.253	

As discussed in Section 4 of [13], 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, 0.7879, 0.7984, 0.8067, 0.8091 for 10%, 20%, 30%, 40%, 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 ≈ 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 32log(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, log(2,000)=11, …, log(5,000)=13 for 1,000, 2,000, …, 5,000 updates, respectively, during the last window). our bit-sequences are shorter (about 80% of Jing's), and yet

achieve much better (or 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 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. Based on the bit-sequence approach [11] and our discussion in [13], we redesigned the hierarchy of the bit-sequences, taking into account the reconnection patterns of mobile clients. We have shown that by using a numerical approximation of integral in solving the nonlinear system, we can derive 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.

8. 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 and T. Imielinski. "Sleepers and workaholics: Caching strategies for mobile environments". *Proc. of the ACM SIGMOD Conference on Management of Data*, pp.1-12, May, 1994.

[4] Christof Bornhövd, Mehmet Altinel, Sailesh Krishnamurthy, C. Mohan, Hamid Pirahesh, Berthold 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 , Shojiro Nishio , Masahiko 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, and 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. K. Elmargarmid, 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, and M. Livny. "Global Memory Management in Client-Server DBMS Architectures". *Proc. of VLDD*, pp. 596-609, August 1992.

[11] C. G. Gray and D. R. Cheriton. "Leases: An Efficient Fault-Tolerant Mechanism for Distributed File Cache Consistency". *Proc. of SOSP*, pp. 202-210, Feb. 1989.

[12] Holliday J, Agrawal, D., El Abbadi, A, "Planned Disconnections for Mobile Databases", Proceedings of 11th International Workshop on Database and Expert Systems Applications, pp. 165 – 169, 2000.

[13] Wen-Chi Hou, Meng Su, Hongyan Zhang, Hong Wang "An Optimal Construction of Invalidation Reports for Mobile Databases" *Proceeding of the 10th International Conference on Information and Knowledge Management (CIKM 2001)* Atlanta, November, 2001.

[14] T. Imielinski, S. Vishwanath, and B. R. Badrinath. "Energy efficient indexing on air", *Proceedings of the ACM SIGMOD Conference on Management of Data*, Minneapolis, Minessota, 1994.

[15] J. Jing, A. K. Elmagarmid, A. Helal, and 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.

[16]Anurag Kahol, Sumit Khurana, Sandeep K. S. Gupta, Pradip K. Srimani:" A Strategy to Manage Cache Consistency in a Disconnected Distributed Environment." IEEE Trans.Parallel Distrib.Syst. 12(7):686-700, 2000.

[17] Kwong Yuen Lai, Zahir Tari, Peter Bertok, "Cost Efficient Broadcast Based Cache Invalidation for Mobile Environments", 18th ACM Symposium on Applied Computing, pp. 871-877, 2003.

[18] Lee, K.C.K, Hong Va Leong, Si, A, "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.

[19] Yan Sheng Lu, Xiong Kai 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.

[20] Pavan Nuggehalli , Vikram Srinivasan , Carla-Fabiana 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.

[21] Xiong-Kai Shao,Yan-Sheng 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.

[22] M., Stonebraker, et al, "Third–Generation Data Base System Manifesto," *SIGMOD Record* 19, 3, pp. 241- 234, Sept. 1990.

[23] 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.

[24] Waluyo, A.B, Srinivasan, B, Taniar, D, "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.

[25] Y. Wang, "Cache Consistency and Concurrency Control in a Client/Server DBMS Architecture", *Proc. of ACM SIGMOD* 1991, pp. 367-376, May, 1991.

[26] Zhijun Wang, Sajal Das, Hao Che and Mohan Kumar,"SACCS: Scalable Asynchronous Cache Consistency Scheme for Mobile Environments", Proc of 23rd International Conference on Distributed Systems Workshop, pp. 797-802, 2003.

[27] K. Wilkinson & M. Neimat, "Maintaining Consistency of Client-Cached Data", *Proc. of* 16th *VLDB*, pp. 122-133, Aug.1990.

[28] K.L. Wu, P.S. Yu and M.S. Chen "Energy-efficient Caching for Wireless Mobile Computing", Proc. 12th *International Conference on Data Engineering*, pp. 34-50, Feb. 1996.