Performance Analysis of Location-Dependent Cache Invalidation Schemes for Mobile Environments

Jianliang Xu, *Member*, *IEEE*, Xueyan Tang, and Dik Lun Lee

Abstract—Mobile location-dependent information services are gaining increasing interest in both academic and industrial communities. In these services, data values depend on their locations. Caching frequently accessed data on mobile clients can help save wireless bandwidth and improve system performance. However, since client location changes constantly, location-dependent data may become obsolete not only due to updates performed on data items but also because of client movements across the network. To the best of the authors' knowledge, previous work on cache invalidation issues focused on data updates only. This paper considers data inconsistency caused by client movements and proposes three location-dependent cache invalidation schemes. The performance for the proposed schemes is investigated by both analytical study and simulation experiments in a scenario where temporal- and location-dependent updates coexist. Both analytical and experimental results show that, in most cases, the proposed methods substantially outperform the NSI scheme, which drops the entire cache contents when hand-off is performed.

Index Terms—Mobile computing, wireless communication, location-dependent information, data management, cache consistency, semantic caching, performance analysis.

1 Introduction

Recent years have witnessed a tremendous proliferation of mobile computing paradigms in both academic and industrial communities. In a mobile computing environment, mobile clients can access a variety of information from anywhere and at any time. Mobility has opened up new classes of applications for mobile computing environments. *Location-dependent information service*, where information provided to users reflects their current locations,[1] is one of these applications which are gaining increasing attention [1], [10], [12], [36]. Through location-dependent information services, mobile clients can access location sensitive data such as traffic reports, travel information, and emergency information. The Advanced Traveler Information Systems ($ATIS$) project has explored this in depth [29]. With continuous development of location identification techniques and growing wireless data rates, location-dependent information service is becoming a reality.

Caching is an important technique to improve system performance in mobile computing environments [4], [18]. However, client locations are not fixed in a mobile environment. The combination of frequent client disconnections and movements makes the design of cache consistency methods a challenge [4], [37]. In recent years, cache consistency in mobile environments has been extensively studied [4], [6], [17], [19]. Most of the previous work however, only studied the cache consistency problem incurred by data updates (hereafter called *temporal-dependent updates*). For location-dependent information in a mobile environment, cache inconsistency can also be caused by location change of a client (hereafter called *location-dependent updates*). Therefore, to maintain cache consistency, cache invalidation should be performed for both temporal-dependent and location-dependent updates. *Temporal-dependent invalidation* and *location-dependent invalidation* affect each other. For example, if the temporal-dependent update rate is much higher than the location-dependent update rate, temporal-dependent updates will dominate cache invalidation, and vice versa.

In a previous paper [37], we proposed three cache invalidation schemes for location-dependent updates, namely *Bit Vector with Compression (BVC)*, *Grouped Bit Vector with Compression (GBVC)*, and *Implicit Scope Information (ISI)*. A preliminary simulation-based study showed that they are able to improve the performance of location-dependent data caching. This paper extends our previous work to analyze the performance of the proposed cache schemes in a scenario where temporal- and location-dependent updates coexist. In the analytical study, we compare the *query cost*s of the proposed schemes with an optimal strategy *OPT*, which assumes perfect location information is available on mobile clients, and a baseline strategy *NSI*, which drops the entire cache contents when hand-off is performed. A series of simulation experiments is also conducted to validate the analysis. Both analytical and experimental results show that the proposed location-dependent invalidation schemes substantially outperform

1. Throughout this paper, it is assumed that a location-dependent query is bound to a mobile client's current location unless explicitly specified. We discuss in Section 7, location-dependent queries bound to other locations.

• *The authors are with the Department of Computer Science, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong. E-mail: {xujl, tangxy, dlee}@cs.ust.hk.*

Manuscript received 15 June 2000; revised 22 Mar. 2001; accepted 13 July 2001.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 112292.

the *NSI* method in most cases. In particular, *ISI* performs very close to the optimal strategy for most system settings, while *GBVC* provides the best scalability.

To the best of our knowledge, studies on location-dependent cache invalidation and its interaction with temporal-dependent cache invalidation do not exist before. The main contributions of our work are summarized as follows:

- introduce the problem of *location-dependent* cache invalidation, which opens up a new direction of research,
- three schemes are proposed to handle the location-dependent cache invalidation issue,
- the performance of the proposed location-dependent invalidation schemes is investigated by analytical study, and the analysis is validated by a series of simulation experiments.

The rest of this paper is organized as follows: Section 2 describes background on potential applications, the location model, the mobile computing model, and temporal-dependent cache invalidation. Section 3 describes the proposed location-dependent invalidation schemes, i.e., *BVC*, *GBVC*, and *ISI*. Section 4 analyzes the performance of the proposed invalidation schemes in a scenario where temporal- and location-dependent updates coexist. The analytical and experimental results are presented in Section 5. Section 6 describes the related work. Finally, Section 7 concludes the paper.

2 Background

In this section, several potential applications for location-dependent information are first described to motivate our study. Afterwards, we provide some background on the location model, the mobile computing model and the temporal-dependent invalidation scheme adopted in this study.

2.1 Potential Applications

Two examples are described below to help us understand potential applications for location-dependent information. Other application scenarios could include community services, health, and entertainment etc., and more can be found on AltaVista Local.[2]

Example 1 (Travel Information). Suppose a traveler is visiting a new city and he wants to find a restaurant around lunch time, but he does not know his current location. The traveler submits a query to the wireless system via his handset, "show me the nearby restaurants." The system responds with one western restaurant. Since he prefers Chinese food, the same query is issued later on while he is visiting the city. After several attempts, the traveler eventually finds the restaurant he likes.

Example 2 (Traffic Information). While traveling in a city, a driver wants to find out the traffic information in his surrounding area from time to time. Thus, he queries the built-in navigation system with "show me the traffic report of nearby area." The system replies immediately. Having this information, he easily optimizes his driving routes.

In both examples, since the user issues the same query repeatedly, the answer might be the same for some consecutive attempts. Caching the query results on mobile clients can save wireless bandwidth and improve system performance.

2.2 Location Model

Location is an important piece of information for representing, storing, and querying location-dependent information. In a location-dependent query, a location is needed to be specified explicitly or implicitly. A location model depends heavily on the underlying location identification technique employed in the system. The available mechanisms for identifying locations can be categorized into two basic approaches:

- **Geometric Model:** A location is specified as a 3-dimensional coordinate, e.g., *GPS* [15] and *Bat* [16]. The main advantage of geometric models is their compatibility across heterogeneous systems. However, because of considerable cost and complexity involved in providing accurate fine-grained location information, the cost/performance ratio of geometric models might not be promising for a large number of applications such as the nearby restaurant example.
- **Symbolic Model:** The location space is divided into disjoint zones and each zone is identified with a unique name. Examples are Cricket [23] and the cellular infrastructure. Being discrete and well-structured, location information based on symbolic models is easier to manage compared to that based on geometric models. For example, location data is much more amenable for database storage and retrieval; they can help analyze location information such as individual mobility patterns.

In this paper, *location granularity* is assumed to be a cell, which falls into the second category. The cell-based location model is well justified with the following two reasons. First, cell-based location identification requires neither additional devices deployed on mobile clients nor modifications over the current cellular network infrastructure. Thus, this is the cheapest solution. Second, with recent development in micro-cell/pico-cell systems,[3] it is believed that this model can provide *precise* location information for most location-dependent information services. The *GUIDE* system [10] is one successful example that benefits from the cell-based location model. On the other hand, although our study assumes a cell-based location model, it is not difficult to see that the proposed techniques can be applied to any other symbolic location models.

2. Website at http://local.altavista.com/.

3. In a typical micro-cell system, the average cell size is of 60-600 m in diameter; in a typical pico-cell system, the average cell size is of 10-60 m in diameter [32].

2.3 Mobile Computing Model

The mobile computing system model adopted in this paper is similar to that in [4]. It consists of two distinct sets of entities: mobile clients (*MC*) and fixed hosts. Some of the fixed hosts, called mobile support stations (*MSS*), are augmented with wireless interfaces. An *MSS* can communicate with the *MC*s within its radio coverage area called a wireless cell. An *MC* can communicate with a fixed host/server via an *MSS* over a wireless channel. The wireless channel is logically separated into two subchannels: an *uplink channel* and a *downlink channel*. The uplink channel is used by *MC*s to submit queries to the server via an *MSS*, while the downlink channel is used by *MSS*s to forward the answers from the server to a target client. We assume an *MC* contacts only one *MSS* at the same time. Each cell is associated with an ID (CID) for identification purposes. A CID is periodically broadcast to all the *MC*s residing in a corresponding cell.

Location-dependent information is provided by the system, i.e., a data item can show different values for different locations and the answer to a query depends on the location where the query originates. The geographical area covered by the information service consists of a number of cells. As mentioned before, this study assumes that the basic location granularity is a cell. The *valid scope* of an item value is defined as the set of cells within which the item value is valid. Since an item may have different values in different cells, an item is associated with a set of valid scopes, which is called the *scope distribution* of the item. To illustrate, let's consider a four-cell system. Suppose that the nearby restaurant for cell 1 and cell 2 is value A and the nearby restaurant for cell 3 and cell 4 is value B. Then, the valid scope of A is $\{1, 2\}$, the valid scope of B is $\{3, 4\}$, and the scope distribution of the nearby restaurant item is $\{\{1, 2\}, \{3, 4\}\}$. We assume the *MSS* of each cell is logically connected to a data server, which provides location-dependent service for the clients in the cell. It is assumed that every data server keeps a complete copy of the database (i.e., the data items are replicated on all the data servers and may have different values for different data servers). Each data server has a global picture about valid scope distributions of all data items. Data updates are assumed to occur at the server-side only. When a data item value is updated, there is a certain delay involved to maintain consistency among the replicas. For simplicity, the delay is assumed to be negligible.

An *MC* can move from one cell to another (called *hand-off*) while retaining its wireless connection. It may disconnect itself from the server voluntarily (to save power or connection cost) or involuntarily (by failure). An *MC* can cache a portion of the database on its local disk or any storage system that survives power-off. It is assumed that a client cache is logically organized and each cache entry contains a data item ID, attached validity information if any (see Section 3 for details), and a pointer pointing to the real data. When a data item is updated at the server-side, the cached copy becomes obsolete. When an *MC* hand-offs to another cell, a cached data item may also become obsolete due to the change of client location, depending on its valid scope.

2.4 Temporal-Dependent Invalidation

Periodical broadcast of invalidation reports (*IR*s) to mobile clients is an efficient strategy to maintain temporal-dependent cache consistency [4]. We assume in this study that a simple strategy *TS* [4] is used for temporal-dependent invalidation. Let L be the *IR* broadcast interval and w the invalidation broadcast window. In the *TS* strategy, an *IR* consists of the current timestamp T and a list (d_i, t_i), where d_i is a data item ID, t_i is the most recent update timestamp of d_i such that $t_i > (T - w \times L)$. In other words, an *IR* contains the update history of the past w broadcast intervals. The timestamp of the latest *IR* received by a client (denoted as T_l) is maintained in its cache. Every mobile client, if active, listens to the report and updates the status of its cache accordingly. If the difference between T and T_l is larger than $w \times L$, the entire cache is dropped. Otherwise, if an item is reported to have changed at a time larger than the timestamp stored in the cache, the *MC* purges it from the cache.

3 Location-Dependent Cache Invalidation Methods

In this section, we describe three proposed cache schemes for location-dependent updates. Since the server generally does not know which items are cached in each client and where each client resides, it is more suitable for clients to initiate the validity checking procedure [4]. In order to validate the cached data, a mobile client can send the IDs of the cached items uplink to the data server. The data server then decides whether these items are still valid according to the mobile client's current location and sends the result back to the client. This method is simple, but it increases network traffic substantially due to the amount of checking involved.

To achieve better performance, the idea is to make use of *validity information* of data items. Specifically, the server delivers the valid scope along with a data item value to a mobile client and the client caches the data as well as its valid scope for later validity checking. This strategy involves two issues, namely validity checking time and validity information organization, that need to be addressed. Since a query result only depends on the location specified with the query, we propose to do validity checking for a cached data value until it is queried. Furthermore, with this strategy a data item value that is invalid with respect to the current location is not necessarily removed from the cache immediately as it may become valid again later. For validity information organization, we propose three methods, i.e., *BVC*, *GBVC*, and *ISI*, in the following sections.

3.1 Bit Vector with Compression (BVC)

In the *BVC* method, the *complete* validity information is attached to a data item value. That is, the complete set of cells in which the data value is valid is kept in the cache. Recall that every cell is associated with a CID to distinguish it from the others. BVC uses a bit vector (BV), corresponding to all the cells to record the valid scope. Obviously, the length of a BV is equal to the number of cells in the system. A "1" in the nth bit

CID	1	2	3	4	5	6	7	8	9	10	11	12
(SDN)Scope Distribution #1	1	2	3	4	5	6	7	8	9	10	11	12
(SDN)Scope Distribution #2	1			2			3			4		
(SDN)Scope Distribution #3	1			2			3			4		5

Fig. 1. An example of data items with different distributions.

indicates that the data item value is valid in the nth cell while "0" means it is invalid in the nth cell.

For example, if there are 12 cells in the system, then a BV with 12 bits is constructed for each cached data item value. If the BV for a data item value is 000000111000, it means this value is valid in the seventh, eighth, and ninth cells only.

The validity checking algorithm then works as follows: Whenever a data item value is required for location-dependent validation, the client listens to the broadcast for the current cell's ID, CID_c, and uses it to examine the cached BV of that item value. If the CID_cth bit is "1" in the BV, it is valid; and otherwise invalid.

It is obvious that with *BVC* the overhead would be significant when the system is large. With the locality of a valid scope, it is possible to perform compression on a *BV* in a real life application.

3.2 Grouped Bit Vector with Compression $(GBVC)$

To remedy the large overhead in *BVC*, the $GBVC$ method keeps track of the validity information of each data item value only for some of the adjacent cells to reduce overhead. The motivation is two-fold. First, a mobile client may seldom move to a cell that is far away from its home area. Consequently, it is enough for a client to know the validity of a data value in the current and neighboring cells. Second, even if a mobile client moves to a distant cell, it takes quite some time for it to do so. During this period of time, the data may have already been updated on the data server and, therefore, the complete validity information of a data item value in distant cells is useless.

Under the $GBVC$ scheme, the whole geographical area is divided into disjoint districts and all the cells within a district form a group. A CID, denoted by (*group-ID*, *intra-group-ID*), consists of a group ID and a cell ID within the group. Validity information attached to a cached data value is represented as a vector of the form (*group-ID*, *BV*) and includes the current *group-ID* and a BV which corresponds to all the cells within the current group. Note that, while delivering a data value to a client, only the BV is attached since the *group-ID* can be inferred from the current CID.

With the same example used in the previous section, suppose that the whole geographical area is further divided into two groups, such that cells 1-6 form group 0 and the rest form group 1. With the $GBVC$ method, one bit is used to construct *group-ID* and a six-bit BV is used to record the cells in each group. For the data item value mentioned earlier, in group 0, the attached bit vector is $(0, 000000)$; in group 1, the attached bit vector is $(1, 111000)$. As can be seen, compared with the BVC method, the overhead for scope information is reduced in the $GBVC$ method.

When a mobile client checks the validity of a data item value, it listens for the current cell's ID, i.e., (*group-ID*$_c$, *intra-group-ID*$_c$), and compares *group-ID*$_c$ with the one associated with the cached data. If they are not the same, the data is invalid. Otherwise, the client checks the *intra-group-ID*$_c$th bit in the BV to determine whether the cached data is valid.

Note that the group size is crucial to the performance of this strategy. If the group is too large, the overhead for maintaining validity information is still very significant. On the other hand, if the group is too small, the chance of mistaking a valid data as invalid might be high since a data item value is regarded as invalid if it is outside its original group. A method to analyze the optimal group size for this scheme is discussed in Section 5.1.

3.3 Implicit Scope Information (ISI)

In BVC, a client stores in the cache the complete validity information of each cached data in the form of a bit vector. The disadvantage of this method is that the size of the validity information could be very large, especially when the system consists of a large number of cells. Consequently, a large bandwidth and cache memory are needed. The advantage is that the validation process is very simple; only the current cell ID is needed. $GBVC$ attempts to reduce the size of the validity information by only keeping partial information in the cache.

The ISI method attempts the other direction by trying to minimize the size of validity information at the expense of the validation procedure. Under this scheme, the server enumerates the scope distributions of all items and numbers them sequentially. The valid scopes within a scope distribution are also numbered sequentially. For any value of data item i, its valid scope is specified by a 2-tuple (SDN_i, SN_i), where SDN_i is the scope distribution number and SN_i denotes the scope number within this distribution. The 2-tuple is attached to a data item value as its valid scope. For example, suppose there are three different scope distributions (see Fig. 1) and data item 4 has distribution 3. If item 4 is cached from cell 6 (i.e., $CID = 6$), then $SDN_4 = 3$ and $SN_4 = 3$. That implies that the cached item 4's value is valid in cells 6 and 7 only.

It can be observed that the size of the validity information for an item value is small and independent of the actual number of cells in which the value is valid. Another observation is that a set of data items may share the same scope distribution. As such, the number of scope distributions could be much smaller than the number of items in the database.

At the server-side, a *location-dependent IR* is periodically broadcast. It consists of the ordered valid scope numbers (SN) in a cell for each scope distribution. For example, in cell 8, the server broadcasts $\{8, 3, 4\}$ to mobile clients, where

TABLE 1
Summary of Notations

Notation	Definition
C	number of cells covered by the service
D	database size in the system
S	number of scope distributions in the database
I	length of a client's residence period
L	length of an IR's broadcast interval
w	window size of the TS report
m	the ratio of I to L, i.e., $m = I/L$
s	sleep probability of client in a broadcast interval
p	probability of remaining in the same cell after a residence period
λ	data query rate for the data item
μ	data update rate for the data item
d	size of the data item
k	valid radius of the data item
g	group radius in the $GBVC$ scheme
$qcost$	query cost for the data item
h	average cache hit ratio for the item
b_a	size of the query's answer (including data and validity information)
b_q	size of an uplink query message
b_t	amortized overheads for temporal-dependent IR broadcasts
b_l	amortized overheads for location-dependent IR broadcasts
b_T	size of a timestamp in the IR
q_o	probability of being awake and no queries in an IR broadcast interval
p_o	probability of no queries in an IR broadcast interval
u_o	probability of no updates in an IR broadcast interval
r_o	probability of no queries being after a random point in an interval
$p_s(i, w)$	probability of having no queries but a "sleeping" streak of w or more intervals in a period of $i - 1$ successive broadcast intervals
$p_v(i, k)$	probability of remaining in the same cluster, with a radius of k, after i residence periods

the three numbers are the SN values in cell 8 for scope distributions 1, 2, and 3, respectively (Fig. 1).

The validity checking algorithm works as follows: After retrieving a location-dependent *IR* for item i, the client compares the cached SN_i with the SDN_ith SN in the location-dependent *IR* received. If they are the same, the cached item value is valid. Otherwise, the data item value is invalid. For example, in cell 8, the client checks for the cached data item 4 whose $SDN_4 = 3$ and $SN_4 = 3$. In the broadcast report, the SDN_4th (i.e., third) SN equals to 4. Therefore, the client knows that data item 4's value is invalid.

4 Performance Analysis

In this section, we analyze the performance for the proposed location-dependent invalidation schemes in a scenario where temporal- and location-dependent updates coexist. The performance metric used in this study is *query cost*, which is defined as the average communication costs (including uplink and downlink costs) for one query. The notations used in the analysis below are summarized in Table 1.

To facilitate our analysis, it is assumed that the system under consideration consists of C hexagonal cells (see Fig. 2a). A random walk pattern is used to model client mobility. It has been argued that pedestrian movements over micro-cells are well captured by the random walk model in a metropolitan area [20]. For simplicity, we assume after a mobile client resides in a cell for a period of I time (called *residence period*), it has a probability of p of remaining in the current cell and uniform probability of moving to any one of its six neighboring cells, i.e., with the same probability $\frac{1-p}{6}$ each (see Fig. 2b).

The database consists of D data items with sizes $d_1, d_2, \cdots, d_D$. For simplicity, valid scopes of data item i are represented as a set of circles, each with a *valid radius* of k_i cells. For example, Fig. 2a shows seven valid scopes, each with a valid radius of one cell. We assume there are totally S different scope distributions in the system, denoted by $s_1, s_2, \cdots, s_S$, where $S \leq D$. To maintain temporal-dependent cache consistency, the server broadcasts an *IR* every L seconds and the TS method [4] is used in our analysis.[4] To simplify the analysis, we assume $L = \frac{I}{m}$, where m is an integer. In TS, the broadcast window size is set to w. We

4. Similar analysis could be applied to scenarios where other temporal-dependent invalidation schemes, such as *AT* and *SIG* [4], are employed.

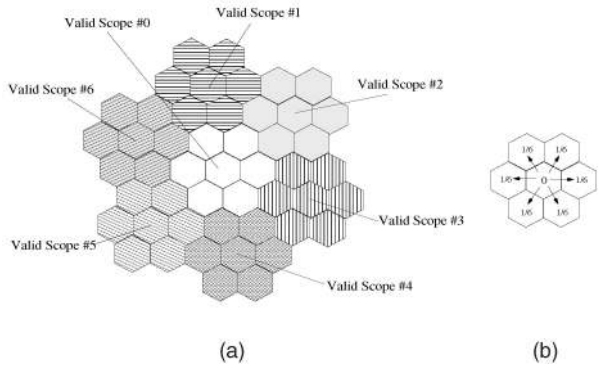


Fig. 2. The structure of the cellular network.

assume a location-dependent *IR*, in *ISI*, is broadcast to mobile clients along with a *TS IR*.

For each item, data queries and updates follow exponential distributions with rates of λ and μ, respectively. Data updates are assumed to happen simultaneously for all values of the same item. We assume only the recently used version of a data item is kept in the cache. If a query has a hit in client cache,[5] it is answered locally. Otherwise, the client goes uplink to get the data. Mobile clients get disconnected and reconnected while they are moving across cells. As in [4], it is assumed that in each IR broadcast interval, mobile clients have a probability s of being disconnected and $1-s$ of being connected, independent of the status in the previous interval.

In what follows, we first introduce several basic parameters used in the analysis. Then, we investigate the performance of the proposed location-dependent schemes (i.e., *BVC*, *GBVC*, and *ISI*) and two additional yardstick schemes (i.e., *OPT* and *NSI*). In the following analysis, a data item has a size of d and a valid radius of k cells.

4.1 Basic Parameters

The query cost, $qcost$, for a data item is:

$$qcost = (1-h)(b_a + b_q) + b_t + b_l, \tag{1}$$

where h is the average hit ratio for the item in client cache, b_a is the size of the query's answer (including the data content and the validity information) on the downlink channel, b_q is the size of an uplink query message, b_t and b_l are the amortized overheads on the downlink channel for maintaining temporal-dependent and location-dependent cache consistency, respectively. Among these terms, the values of $h, b_a,$ and b_l depend on the location-dependent cache schemes and we will analyze them in the following sections; b_q is a system parameter; and b_t can be obtained as follows: Let b_T denote the size of each timestamp in the TS reports. The average TS report size is $D(1-e^{-\mu w L})(logD + b_T)$ [4]. Since an IR is broadcast every L seconds and the overhead is amortized to each query, b_t can be calculated using the following equation:

$$b_t = \frac{D(1-e^{-\mu w L})(logD+b_T)}{D\lambda L} = \frac{(1-e^{-\mu w L})(logD+b_T)}{\lambda L}. \tag{2}$$

In the following, we introduce several basic parameters for calculating $h, b_a,$ and b_l. It is easy to obtain [4]:

$$Prob[\text{no queries in an } IR \text{ broadcast interval} \mid \text{client is awake during the interval}] = e^{-\lambda L}; \tag{3}$$

$$q_o = Prob[\text{awake and no queries in an } IR \text{ broadcast interval}] = (1-s)e^{-\lambda L}; \tag{4}$$

$$p_o = Prob[\text{no queries in an } IR \text{ broadcast interval}] = s + q_o; \tag{5}$$

$$u_o = Prob[\text{no updates in an } IR \text{ broadcast interval}] = e^{-\mu L}. \tag{6}$$

Given a random point in a broadcast interval, denote r_o as the probability of having no queries in the interval after the point. Then, we have

$$r_o = \frac{1}{L}\int_{t=0}^{L} e^{-\lambda(L-t)}dt = \frac{1}{\lambda L}(1-e^{-\lambda L}). \tag{7}$$

Consider a period of $i-1$ successive intervals, let $p_s(i,w)$ be the probability of having no queries, but a "sleeping" streak of w or more intervals in this period. We get the following equation:[6]

$$p_s(i,w) = \begin{cases} 0 & \text{if } i \le w \\ s^w & \text{if } i = w+1 \\ s^w p_o^{i-w-1} + \sum_{j=0}^{i-w-2}(p_o^j - p_s(j,w))q_o s^w p_o^{i-j-w-2} & \text{if } i > w+1 \end{cases} \tag{8}$$

The detailed analysis is given in Appendix A.

Further, let $p_v(i,k)$ denote the probability for a client to remain in the same cluster, which has a radius of k cells, after i residence periods. The derivation of $p_v(i,k)$ is shown in Appendix B.

4.2 Optimal Query Cost

Let's first consider an ideal strategy OPT, where mobile clients have perfect knowledge about valid scopes of cached data. This strategy allows clients to do location-dependent validity checking correctly without introducing any cost, i.e., $b_l = 0$ and $b_a = d$. Under this strategy, the average cache hit ratio is maximized, i.e., a cached data item is considered invalid if and only if it is invalid. Denote the maximal hit ratio as h_{max}, and we have

5. In order to concentrate on the cache consistency issue, we assume client cache is of infinite capacity in most of this study. Under this assumption, a cache miss is only due to data updates or client movements. The case of finite cache capacity is investigated in Section 5.5.

6. No solution for $p_s(i,w)$ is given in [4], only an upper bound and a lower bound were provided.

$$h_{max} = (1 - r_o) + r_o \sum_{i=1}^{\infty} \left[(1 - p_o)(p_o^{i-1} - p_s(i, w)) u_o^i \cdot \frac{1}{m} \sum_{j=0}^{m-1} p_v((i + j)/m, k) \right]. \quad (9)$$

The detailed derivation of h_{max} is given in Appendix C. We can then obtain the optimal query cost

$$qcost_{OPT} = (1 - h_{max})(d + b_q) + b_t. \quad (10)$$

4.3 NSI

No Scope Information (NSI) denotes a strategy in which mobile clients do not have validity information about cached data and drop the cache contents entirely at handoff. This strategy is a practical scheme and is used as a yardstick in our performance evaluation. Since NSI does not need validity information as in OPT, $b_l = 0$, and $b_a = d$. We now derive its average cache hit ratio. A cached data item is hit only when the client remains in the same cell between two consecutive queries. Therefore, similar to the derivation of h_{max}, the hit ratio of *NSI* is given by:

$$h_{NSI} = (1 - r_o) + r_o \sum_{i=1}^{\infty} \left[(1 - p_o)(p_o^{i-1} - p_s(i, w)) u_o^i \cdot \frac{1}{m} \sum_{j=0}^{m-1} p^{(i+j)/m} \right]. \quad (11)$$

Thus, the query cost can be obtained:

$$qcost_{NSI} = (1 - h_{NSI})(d + b_q) + b_t. \quad (12)$$

4.4 BVC

Since BVC caches complete validity information in the form of a bit vector, the overhead of validity information for each item value is C bits, where C is the number of cells in the system. Thus, $b_a = d + C$. Since there is no need to broadcast any extra information for maintaining location-dependent cache consistency, $b_l = 0$. Furthermore, $h = h_{max}$ because a client has complete information about a data value's valid scope in the BVC scheme. Hence, the query cost of BVC is given by:

$$qcost_{BVC} = (1 - h_{max})(d + C + b_q) + b_t. \quad (13)$$

4.5 GBVC

In *GBVC*, we assume a group is defined by a circle. Consider a group with a radius of g cells. The overhead of validity information for each item value is $3g^2 + 3g + 1$. Thus, $b_a = d + 3g^2 + 3g + 1$. Similar to BVC, $b_l = 0$. To simplify the computation of the average hit ratio, we ignore the mismatch between boundaries of groups and valid scopes in the analysis. We will investigate such inaccuracy in Section 5. Notice that, under the *GBVC* scheme, a cached data item is regarded as invalid if the client moves out of its valid scope or the local group. Therefore, the cache hit ratio is given by:

TABLE 2
Default System Parameter Settings

Para.	Setting	Para.	Setting
C	10000	D	10000
S	100	I	60 seconds
L	2 seconds	w	10
s	0.2	p	0.8
λ	0.1	μ	0.001
d	4096 bits	k	10
b_q	32 bits	b_I	32 bits

$$h_{GBVC} = (1 - r_o) + r_o \sum_{i=1}^{\infty} \left[(1 - p_o)(p_o^{i-1} - p_s(i, w)) u_o^i \cdot \frac{1}{m} \sum_{j=0}^{m-1} p_v((i + j)/m, min(g, k)) \right]. \quad (14)$$

Thus, we obtain the query cost of $GBVC$ as:

$$qcost_{GBVC} = (1 - h_{GBVC})(d + 3g^2 + 3g + 1 + b_q) + b_t. \quad (15)$$

4.6 ISI

In *ISI*, the overhead of validity information consists of a scope distribution number and a scope number for any item value. For a scope distribution with a valid radius of k cells, the scope number is approximated as $\frac{C}{3k^2+3k+1}$. Thus,

$$b_a = d + logS + log\left(\frac{C}{3k^2 + 3k + 1}\right),$$

where the second term is the size of a scope distribution number and the third term is the size of a scope number. Under the ISI scheme, the broadcast location-dependent IR consists of S scope numbers. Let k_{s_j} denote the valid radius of the data items in scope distribution s_j, $j = 1, 2, \cdots, S$. Since the overhead of a report is amortized to each query, we get b_l as:

$$b_l = \frac{\sum_{j=1}^{S} log\left(\frac{C}{3k_{s_j}{}^2 + 3k_{s_j} + 1}\right)}{D\lambda L}. \quad (16)$$

As in *BVC*, $h = h_{max}$, and the query cost is given by:

$$qcost_{ISI} = (1 - h_{max})\left(d + logS + log\left(\frac{C}{3k^2 + 3k + 1}\right) + b_q\right) + b_t + \frac{\sum_{j=1}^{S} log\left(\frac{C}{3k_{s_j}{}^2 + 3k_{s_j} + 1}\right)}{D\lambda L}. \quad (17)$$

5 Numerical Results

This section presents the numerical results obtained through analysis and simulation. They serve two purposes: 1) to validate the analysis in Section 4 and 2) to evaluate the proposed location-dependent invalidation schemes under various system configurations. The equations developed in Section 4 are used to calculate the analytical results and the simulation is implemented with CSIM [28]. Table 2 shows the default system settings (as in Section 4, λ and μ are for a single data item).

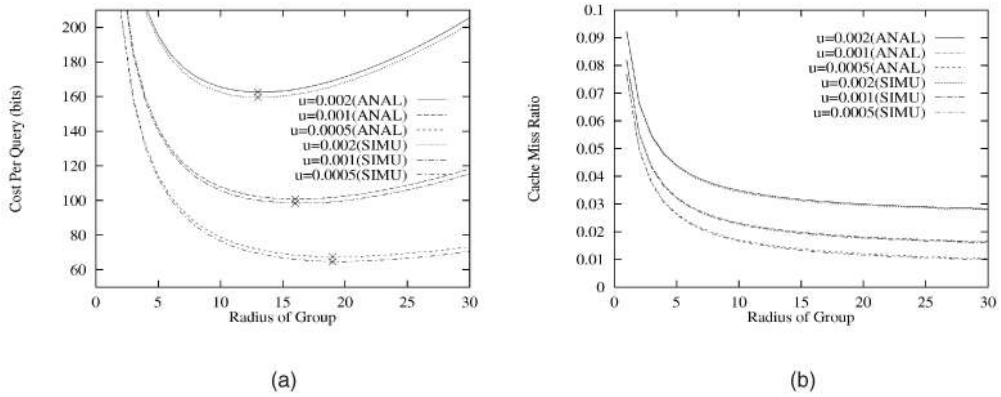


Fig. 3. Performance under various group sizes versus update rates (μ). (a) Query cost performance. (b) Cache miss ratio performance.

In the following sections, the optimal group size for the *GBVC* scheme is first examined. Next, the performance of the proposed location-dependent invalidation schemes is evaluated under various valid scope sizes. Then, the effect of different update rates and movement patterns on the proposed schemes is studied. After that, the scalability of the proposed schemes is assessed. Finally, the proposed schemes are evaluated under finite cache capacity.

5.1 Analysis of the Optimal Group Radius for *GBVC*

As mentioned in Section 3.2, the performance of *GBVC* depends heavily on the group size. In this section, we analyze the optimal group size for *GBVC* using the following method: We assume an item shows only one value and its valid scope covers all the cells (i.e., k is infinity). We then vary the group radius from one cell to a sufficiently large number of cells (30 used in our simulation) exhaustively, and find out the optimal group size.

Fig. 3 shows the performance of *GBVC* under three different data update rates, where *ANAL* and *SIMU* represent the results obtained through analysis and simulation, respectively.[7] It can be seen that the analytical results are very close to the corresponding simulation results.

In Fig. 3a, the query cost performance of *GBVC* improves first, and degrades then. This could be explained as follows: Query cost is determined by both cache miss ratio and system overhead for maintaining validity information. The choice of group size reflects the tradeoff between these two factors. As group size increases, a client has more validity information about the cached data. Therefore, the chance of regarding a valid data as invalid decreases and the cache miss ratio is improved (see Fig. 3b). On the other hand, the larger the group size, the more the overhead of validity information. As a result, there exists an optimal group radius where the query cost is minimized.

From Fig. 3a, the performance of *GBVC* under data update rates of 0.002, 0.001, and 0.0005 is optimal when the group radii are 13, 15, and 19 cells, respectively. The result is consistent in both analysis and simulation. We observe that the optimal group size is increased as the data update rate decreases. The reason is as follows: As can be observed in Fig. 3b, as the group radius increases, a lower update rate has a higher relative improvement in cache miss ratio in terms of percentage. For example, when the group radius is increased from 5 to 15, the improvement in cache miss ratio is 48 percent for $\mu = 0.0005$, 39 percent for $\mu = 0.001$, and 28 percent for $\mu = 0.002$. Since the query cost is the product of cache miss ratio and size of a query result, a higher relative improvement in cache miss ratio allows more validity information (i.e., a larger group) to be attached before reaching the optimal point. Consequently, the optimal group size obtained for a lower update rate becomes larger.

Fig. 4 shows the performance of *GBVC* under three different cell residence periods when the group radius is varied. Again, the analytical results are consistent with the simulation results. The optimal group radii for residence periods of 30, 60, and 120 are 19, 15, and 13 cells, respectively. With increasing cell residence period, the optimal group size is decreased. The reason is as follows: For a longer residence period, a smaller group size would achieve a performance close to the minimal cache miss ratio (see Fig. 4b). As a result, when the group size increases, a longer residence period has a less relative improvement in cache miss ratio in terms of percentage. Thus, the optimal point becomes smaller for a longer residence time.

In subsequent sections, the *GBVC* scheme takes the optimal group size obtained using the analysis technique presented in this section.

5.2 Performance for Varying Valid Scope Size

This section evaluates the performance of the cache invalidation schemes under various valid scope sizes. Fig. 5 shows the results when the valid radius is varied from 0 to 25. Since the analytical results of cache hit ratio for *OPT*, *BVC*, and *ISI* are the same (refer to Section 4), a single curve is used to represent all of them.

From Fig. 5, it can be observed that the analytical results for all the schemes are very close to the corresponding experimental results. This validates the analysis performed in Section 4. For *GBVC*, the analytical query costs are slightly smaller than the experimental ones in most cases. Such inaccuracy is due to the fact that the mismatch between boundaries of groups and valid scopes is ignored in the analysis. For valid radii of 0 and 15 cells, the analytical and experimental results of *GBVC* are almost the same. This is because the boundaries of groups and valid scopes match perfectly in these cases.[8] We have conducted experiments with other system settings, it is also observed

7. This also applies to all the figures shown afterwards.

8. In our simulation, the group and valid scope distribution patterns are assumed to be the same if they have the same radius. The optimal group size for *GBVC* in this set of parameter settings is 15.

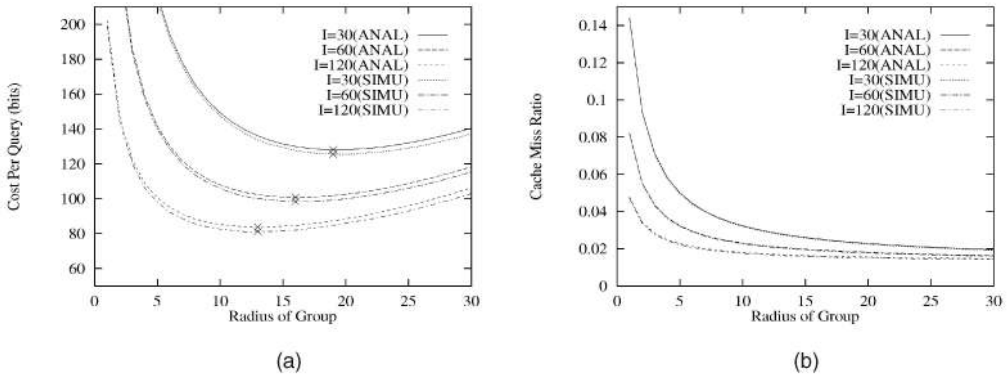


Fig. 4. Performance under various group sizes versus residence period (I). (a) Query cost performance. (b) Cache miss ratio performance.

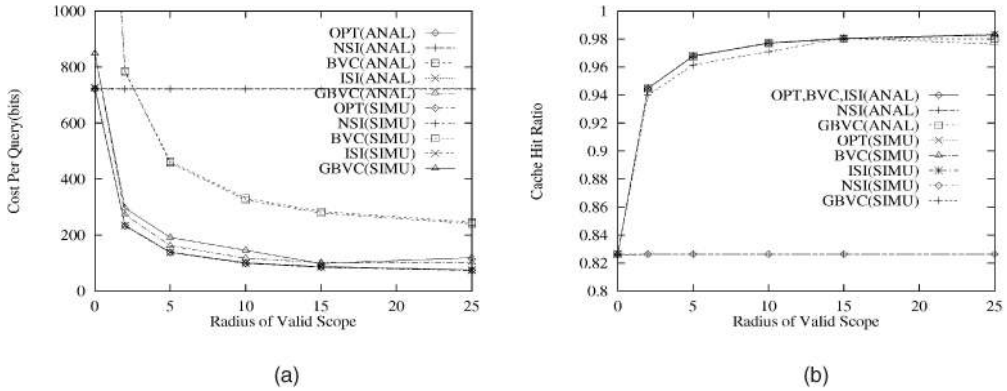


Fig. 5. Performance under various scope sizes. (a) Quercy cost performance. (b) Cache hit ratio performance.

that the analytical results approach the simulation results pretty well for all the schemes except *GBVC*. Thus, to improve the clarity of the figures, the analytical and experimental results for the schemes other than *GBVC* are merged into one curve in the figures shown afterwards.

In Fig. 5a, we can see that the proposed location-dependent invalidation schemes reduce query cost substantially over *NSI* in most cases. For example, when the valid radius is greater than 0, query costs of *ISI* and *GBVC* are 60 percent to 85 percent lower than that of *NSI*. The performance improvement of the proposed schemes becomes more pronounced with increasing valid scope size. When the valid radius becomes larger than 10, all the curves begin to become flat as the optimal cache hit ratio cannot be further improved a lot after that.

Among the proposed schemes, *ISI* shows the best performance (overlapped with *OPT* in Figs. 5a and 5b); *GBVC* is the second, followed by *BVC*. The query costs of *ISI* and *GBVC* approach that of the optimal strategy *OPT*. Since all the schemes have similar cache hit ratio performance as shown in Fig. 5b, the performance difference between the proposed schemes is mainly due to the different amount of overhead introduced for maintaining validity information. Examine the case where the valid radius is 0. *ISI* has query cost similar to *OPT*, *GBVC* is 17 percent worse than *ISI*, and *BVC* has a very bad performance (2462.84 bits, not plotted in Fig. 5a, 240 percent worse than *ISI*). In this case, all proposed schemes have the same cache hit ratio as *OPT* (Fig. 5b) and, hence, the difference in query cost can be directly attributed to the different amount of overhead. Thus, the results imply that the overhead for maintaining scope information is trivial in *ISI* but significant in *BVC*.

5.3 Effect of the Update and Movement Patterns

As mentioned earlier, temporal-dependent and location-dependent invalidations affect each other. In this set of experiments, we study the performance trends for different data update rates and client movement rates by fixing one parameter and varying the other.

Fig. 6 shows the results as the data update rate decreases from 0.1 to 0.0001. As expected, all the schemes perform better when the update rate becomes lower. When the update rate is 0.1, temporal-dependent invalidation occurs very frequently. Thus, even if a client preserves valid items in the new cell after hand-off, these items would be purged very quickly due to frequent temporal-dependent invalidation. Location-dependent invalidation, in this case, is not a dominant factor in cache invalidation. Therefore, *ISI* and *GBVC* perform about the same as *NSI*, *BVC* is even worse than *NSI* due to a large overhead. As the update rate decreases, *ISI* and *GBVC* outperform *NSI* more significantly; *BVC* becomes superior over *NSI* after $\mu = 0.001$. The reason is that, for a lower update rate data items retained after hand-off using the proposed schemes become more useful and, hence, improve the cache hit ratio over *NSI* more greatly (see Fig. 6b).

Fig. 7 shows the results as the residence period varies from $6s$ to $6{,}000s$. The longer the residence period, the less frequent the client movement. As the residence period increases, all the schemes achieve a better performance due to less location-dependent invalidation. When the

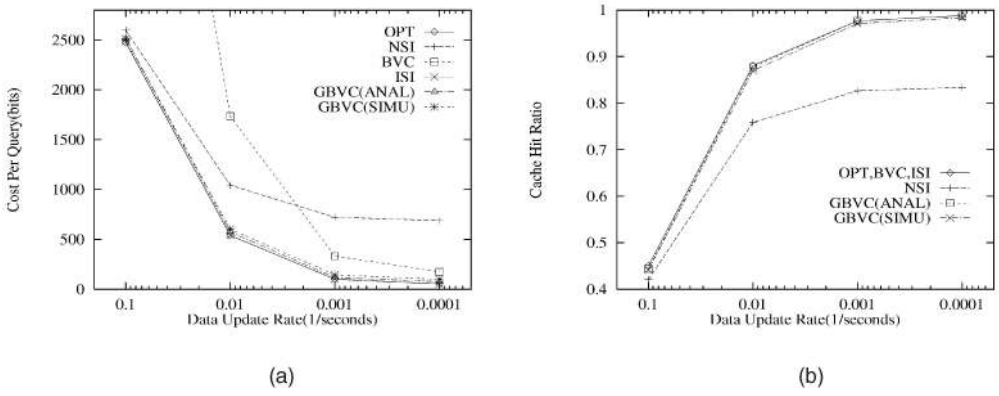


Fig. 6. Performance under different data update rates. (a) Query cost performance. (b) Cache hit ratio performance.

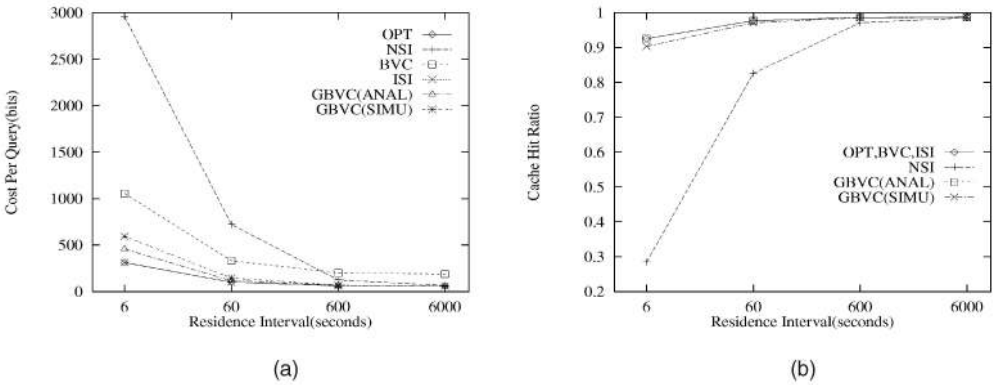


Fig. 7. Performance under different residence intervals. (a) Query cost performance. (b) Cache hit ratio performance.

movement occurs frequently (left part of the figures), the improvement of the proposed schemes in cache hit ratio and, hence, query cost over *NSI* is tremendous since these schemes to various degrees can prevent cached items from being purged during hand-off. As the residence period increases, the performance improvement of the proposed schemes becomes less significant. When the residence period reaches 6,000s, *ISI* and *GBVC* have a similar performance to *NSI*, while *BVC* performs worse than *NSI*. This reason is as follows: With an increasing residence period, the chance for a client's residing in the same cell between two consecutive queries becomes higher, thus the performance difference between the proposed schemes and *NSI* becomes less in terms of cache hit ratio as well as query cost. Because of a large overhead for maintaining validity information, *BVC* has a worse performance than *NSI* when I is larger than 600s.

5.4 Scalability

We now study the scalability of the proposed schemes. It is observed in the previous sections that *BVC* does not perform as good as *GBVC* and *ISI* due to high overhead. In fact, the overhead of *BVC* is directly proportional to the number of cells in the system. As shown in Fig. 8, the performance of the *BVC* scheme degrades significantly when the system contains a large number of cells, while the performance of the other schemes remains quite stable. Therefore, *BVC* is not scalable to the number of cells in the system.

We also notice that the query cost of the *ISI* scheme depends on the number of scope distributions (S) in the database. For a larger S, a larger location-dependent IR needs to be broadcast. Figs. 9a and 9b show the query costs for a different number of scope distributions when the query rate is set to 0.1 and 0.001, respectively.

As expected, the query cost of *ISI* increases with the number of scope distributions, while all the other schemes are insensitive to the number of scope distributions. Since the cost of broadcasting a location-dependent *IR* is amortized over all queries issued in a broadcast interval, the amount of performance degradation for *ISI* depends on the query rate (λ). When λ is high, the degradation is moderate (see Fig. 9a), and when λ is low, the degradation is considerable (see Fig. 9b).

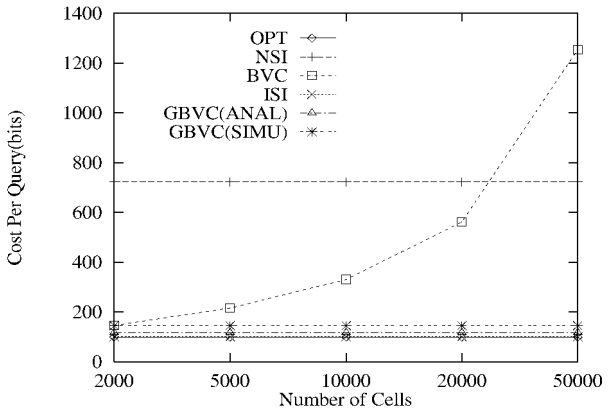


Fig. 8. Scalability of *BVC*.

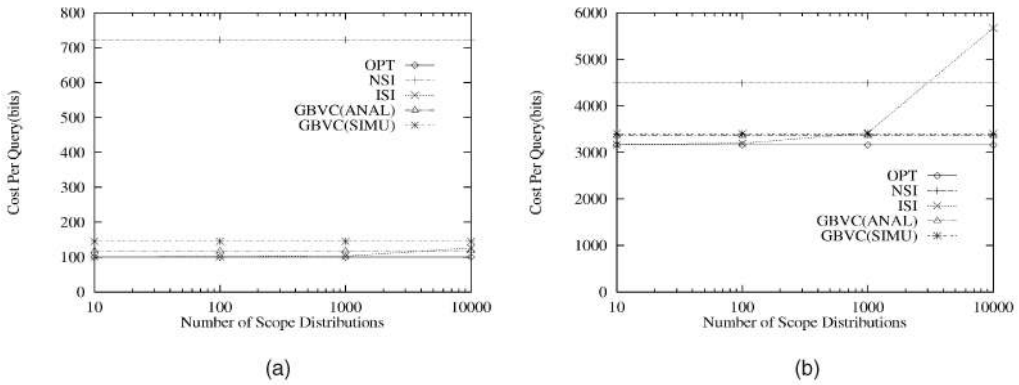


Fig. 9. Scalability of *ISI*. (a) $\lambda = 0.1$. (b) $\lambda = 0.001$.

5.5 Performance Under Finite Cache Capacity

So far, we have evaluated location-dependent cache invalidation schemes under the assumption of infinite cache capacity. This assumption is reasonable when the query popularity distribution of data items is highly skewed, since under this case, a small cache would be enough to cache most frequently accessed data. In this set of experiments, we examine the schemes under finite cache capacity. Client queries and data updates are uniformly distributed among all data items, representing the worst scenario for the cache performance, and their aggregate rates are set to 10 and 0.1, respectively. *LRU* is employed as the cache replacement policy.

As shown in Fig. 10, caching performance of the proposed schemes improves with increasing cache size. However, the performance of *NSI* is quite steady. This indicates that *NSI* cannot fully utilize cache space due to unnecessary location-dependent invalidation. Comparing the performance when cache capacity is finite and infinite, the relative performance of *GBVC*, *ISI*, and *OPT* is similar. This is because *GBVC* and *ISI* incur negligible overhead for storing attached validity information. On the other hand, due to high storage overhead, the hit ratio of *BVC* decreases significantly when cache capacity is finite. We can see that *BVC* performs even worse than *NSI* in terms of query cost in this case.

6 Related Work

Caching is a commonly used technique to improve performance in traditional distributed systems [3], [27] and Web environment [33]. In the context of wireless mobile environments, client data caching is much more desirable due to constraints such as scarce wireless bandwidth and limited power source, etc. However, frequent client disconnections and movements across cells make the design of cache consistency methods a challenge. In recent years, much research effort has been devoted to developing efficient cache consistency protocols [4], [5], [6], [9], [13], [17], [19], [31], [35]. Most of the previous work however, considered temporal-dependent cache consistency only. To our knowledge, no previous work has explored the issue of location-dependent consistency and the combination of temporal- and location-dependent cache invalidations.

Barbara and Imielinski were first to address the cache consistency issue for mobile environments. In [4], repetitive broadcast of invalidation reports (IR) is used to notify mobile clients of the recent updates at the server. Three IR schemes, namely Broadcasting Timestamp (TS), Amnesic Terminals (AT), and Signatures (SIG), were presented. There is a lot of work subsequently, for example [5], [6], [8], [9], [13], [17], [19], [31], [35]. Basically, these papers were devoted to designing efficient algorithms to reduce IR overhead and improve uplink cost. In particular, Jing et al.

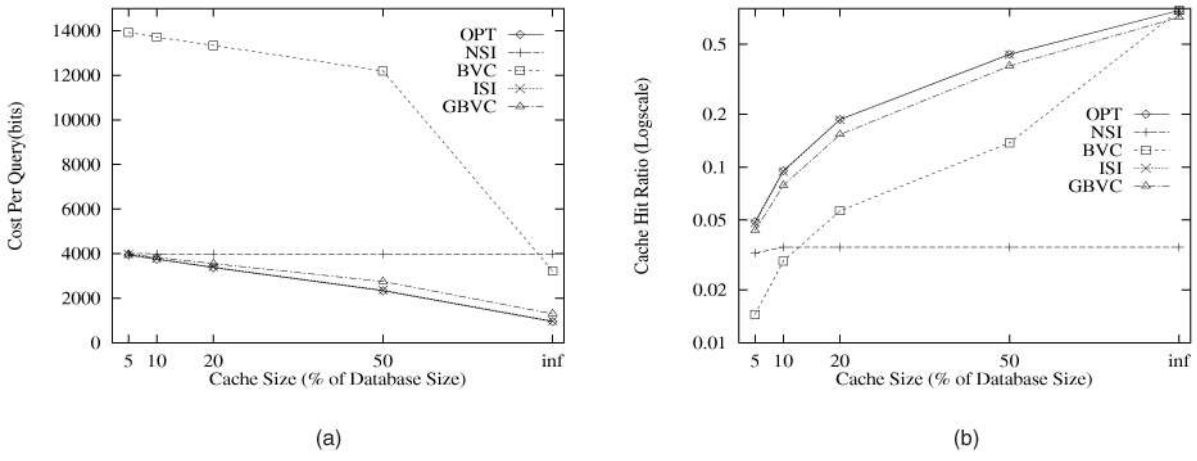


Fig. 10. Performance under finite cache capacity. (a) Query cost performance. (b) Cache hit ratio performance.

proposed a scheme called Bit-Sequence (BS), with a compressed invalidation report which consists of a set of binary bit sequences each with a corresponding timestamp [19]. This algorithm is particularly attractive for mobile clients of frequent disconnections. In [17], Hu et al. introduced several adaptive cache invalidation schemes. By dynamically adjusting the current broadcast window and choosing the algorithm (among TS, AT, and BS) for constructing the next IR, it can adapt to different system workloads while maintaining the optimal overall performance in terms of throughput and power consumption.

Different from the above item-based cache consistency algorithms, Cai et al. introduced a view-based cache consistency algorithm with incremental techniques [7]. Recent work by Pitoura and Chrysanthis examined the cache consistency issue for mobile read-only transactions in broadcast disks [22].

A prefetching technique for context-aware information services was recently discussed by Persone et al. [21]. Based on a proposed cost model, they evaluated some prefetching schemes under different mobility models. However, the valid scope for all data item values was assumed to be the same, i.e., one cell. Thus, the organization of validity information was simplified. Semantic data caching has also been suggested to manage location-dependent query results [11], [24], [25], where a cached result is described with the location(s) associated with the query. Based on the location information, distance-based cache replacement policies were proposed [11], [25]. Unfortunately, they did not explore possible validity of a cached query result with locations different from that associated with the query which, as demonstrated in this paper, could enhance the performance of location-dependent data caching.

Other related work includes the studies on using database technologies to manage moving objects [30]. A moving object continuously changes its location in an unpredictable way. Thus, the major issues arising in this research area are how to model, update, index, and query moving objects [14], [26], [34]. While these studies concern data management for moving object data servers, in this paper, we are interested in how to improve data access performance by employing advanced client caching techniques. The objects queried could be moving or static.

7 Conclusion

In this paper, we have studied the cache consistency issue for location-dependent information in the context of mobile environments. A cache inconsistency issue caused by both temporal- and location-dependent updates has been addressed. For location-dependent updates, three cache invalidation schemes have been proposed. They differ from one another in validity information organization.

The proposed schemes have been investigated by an analytical study in a scenario where temporal- and location-dependent updates coexist. A series of experiments has been conducted to validate the analysis. Both analytical and experimental results show that, in most cases, the proposed schemes are able to achieve a much better performance than the baseline scheme which drops the cache contents entirely at hand-off. Particularly, the ISI scheme is very close to the optimal strategy for a small and medium number of scope distributions. While BVC cannot scale up to system size and ISI does not perform well in a system with a large number of scope distributions and a low query rate, the $GBVC$ method shows the best scalability. Moreover, we observed that when the data update rate is much higher than the client movement rate, sophisticated location-dependent cache invalidation may be unnecessary.

In this paper, we have assumed location-dependent queries are based on the current location of a client. For some applications, the queries may be required to be bound to other locations. Our proposed location-dependent caching schemes are still applicable in these cases with slight modifications. For *BVC* and *GBVC*, a mobile client can simply check data validity with respect to the location specified with the query. For *ISI*, the entire scope distribution table needs to be stored at the clients to translate the attached (SDN_i, SN_i) into CIDs and enable validity checking. The impact of such queries on the performance of the proposed schemes will be investigated in future work.

Location-dependent data caching opens up a new dimension of research in mobile computing. Possible improvement can be made on the proposed cache invalidation schemes. Since mobile clients may have different movement patterns, per-user-based adaptive techniques can be developed. In this paper, we have considered applications based on a symbolic location model, future work is required to consider applications based on a geometric location model. Besides cache invalidation schemes, cache prefetching and replacement policy deserve future work. The following two aspects make them different from the traditional cache schemes: 1) multiple up-to-date versions of a data item can coexist in the cache to improve performance and 2) valid scope and distance are two additional factors needed to take into consideration for the design of a cache scheme. Furthermore, how to incorporate location-dependent data invalidation schemes and semantic caching techniques [25] would be an interesting topic.

Appendix A

Derivation of $p_s(i, w)$

Given a period of $i-1$ successive intervals (denoted by $I_1, I_2, \cdots, I_{i-1}$), the probability of having no queries but a "sleeping" streak of w or more intervals in this period, i.e., $p_s(i, w)$ can be calculated in an iterative way.

Obviously,

$$p_s(i, w) = 0, \quad \text{if } i - 1 < w,$$

and

$$p_s(i, w) = s^w, \quad \text{if } i - 1 = w.$$

Consider the case where $i - 1 > w$. Let $p_s(i, w, j)$ be the probability that the first "sleeping" streak of w or more intervals starts from I_{j+2}. If $j \geq 0$, we have

$$p_s(i, w, j) = (p_o^j - p_s(j, w)) \cdot q_o \cdot s^w \cdot p_o^{i-j-w-2},$$

where $(p_o^j - p_s(w, j))$ is the probability of having no queries and no "sleeping" streak of w or more intervals from I_1 to I_j, q_o is the probability that I_{j+1} is awake and has no queries, s^w is the probability that $I_{j+2}, I_{j+3}, \cdots, I_{j+w+1}$ are all sleeping, and $p_o^{i-j-w-2}$ is the probability of having no queries from I_{j+w+2} to I_{i-1}.

If $j = -1$ (i.e., the first "sleeping" streak starts from I_1), it is easy to obtain

$$p_s(i, w, -1) = s^w p_o^{i-w-1}.$$

Notice that, given i where $i-1>w$, the starting interval of the first "sleeping" streak is likely to be $I_1, I_2, \cdots$, or I_{i-w} (i.e., j can vary from -1 to $i-w-2$). Hence, if $i-1>w$,

$$p_s(i,w) = \sum_{j=-1}^{i-w-2} p_s(i,w,j) = s^w p_o^{i-w-1} + \sum_{j=0}^{i-w-2} (p_o^j - p_s(j,w)) q_o s^w p_o^{i-j-w-2}.$$

Therefore, we get

$$p_s(i,w) = \begin{cases} 0 & \text{if } i \le w \\ s^w & \text{if } i = w+1 \\ s^w p_o^{i-w-1} + \sum_{j=0}^{i-w-2} (p_o^j - p_s(j,w)) q_o s^w p_o^{i-j-w-2} & \text{if } i > w+1 \end{cases}$$

APPENDIX B

DERIVATION OF $p_v(i,k)$

We now derive the probability of a client remaining in the same cluster, which has a radius of k cells, after i residence periods, i.e., $p_v(i,k)$. The cells in the cluster can be classified into several *type*s. Two cells, A and B, are of the same type if the multiset of types for A's neighbors is the same as that for B's neighbors [2].

The type construction algorithm can be found in [2]. Fig. 11 illustrates the cell types for a cluster with a radius of four cells, where all types in this cluster are covered by the shaded cells. Let a type represent a state, and we have the state diagram as shown in Fig. 12.

Let $p_{(x,y)(x',y')}$ be the probability from state (x,y) to state (x',y') in a residence period. The *transition matrix* is obtained as $P = (p_{(x,y)(x',y')})$ (see Fig. 13).

For $i \ge 1$, let

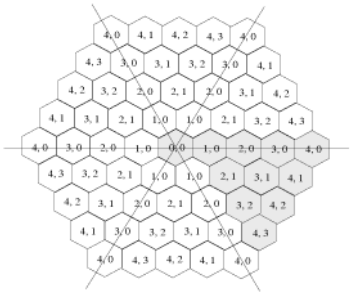


Fig. 11. An example of cell type classification.

$$P^{(i)} = \begin{cases} P & \text{if } i = 1 \\ P \times P^{(i-1)} & \text{if } i > 1 \end{cases} \quad (18)$$

An element $p^{(i)}_{(x,y)(x',y')}$ in $P^{(i)}$ is the probability that a client moves from state (x,y) to state(x',y') with exact i steps. Suppose that a client initially resides at cell $< x,y >$ within the cluster, the probability of its staying in the cluster after i steps could be computed as

$$p_{<x,y>}(i,k) = p^{(i)}_{(x,y)(0,0)} + \sum_{1 \le x' \le k} \sum_{y' \le x'-1} p^{(i)}_{(x,y)(x',y')}. \quad (19)$$

Except for state (0,0), which corresponds to type (0,0), one state/type corresponds to six cells in the real cluster (Fig. 11). Thus, the probability $p_v(i,k)$ is calculated as follows:

$$p_v(i,k) = \frac{p_{<0,0>}(i,k) + 6 \cdot \sum_{1 \le x \le k} \sum_{y \le x-1} p_{<x,y>}(i,k)}{3k^2+3k+1}. \quad (20)$$

APPENDIX C

DERIVATION OF h_{max}

In order to compute the hit ratio h_{max}, we assume a query has occurred at a particular instant of time and we compute

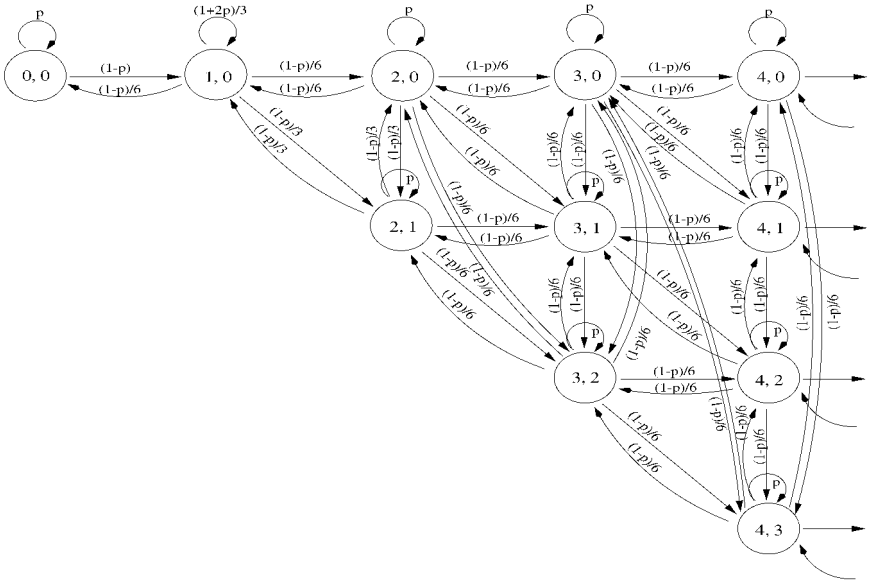


Fig. 12. State transition diagram.

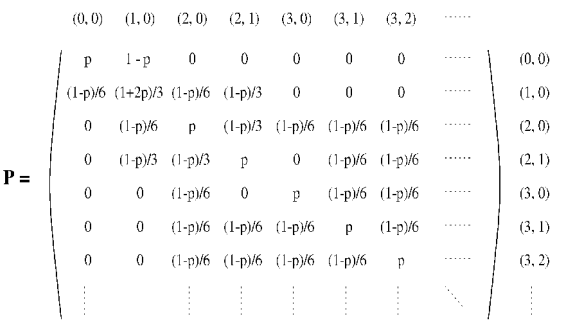


Fig. 13. Transition matrix.

the conditional probability of the cached value being valid in the next query.

First, consider the case where the next query and the current query occur in the same IR broadcast interval. According to the TS invalidation scheme, these two queries cannot be answered until the arrival of the next IR [4]. Thus, actually these two queries are answered at the same time. Therefore, the hit ratio for the next query is 1.

Now consider the case where the next query occurs in the interval of i intervals away from the current interval, $i \geq 1$. The condition for the next query to be a cache hit is that there is no temporal-dependent and location-dependent invalidations during i periods.

For no temporal-dependent invalidations and no queries during i periods, the following two conditions must hold: 1) the data item cannot be updated during i periods and 2) the client cannot sleep for more than w successive intervals during $i-1$ periods. Therefore, the probability of no temporal-dependent invalidations is $u_o^i(p_o^{i-1} - p_s(i,w))$, where the first term is the probability of no updates during i intervals, and the second term is the probability of having no queries and no "sleeping" streak of w or more intervals during $i-1$ intervals.

In order to calculate the probability of no location-dependent invalidations during i periods, we need to know the number of residence periods before the next query. The number of residence periods could be one of the following values, each with the same probability: $i/m, (i+1)/m, \cdots, (i+m-1)/m$. Therefore, we can get the probability of no location-dependent invalidations as $\frac{1}{m}\sum_{j=0}^{m-1} p_v((i+j)/m, k)$, where $p_v(i,k)$ is given by (20).

Thus, we get the probability of a cache hit for the second case as follows:

$$\sum_{i=1}^{\infty}\left[(1-p_o)\cdot u_o^i(p_o^{i-1} - p_s(i,w))\cdot\frac{1}{m}\sum_{j=0}^{m-1} p_v((i+j)/m,k)\right], \tag{21}$$

where the first term is the probability of the next query.

Obviously, the probability of the second case to take place is r_o. Thus, the probability of the first case to take place is $1-r_o$. Combining these two cases, we finally obtain the hit ratio h_{max}:

$$h_{max} = (1-r_o) + r_o\sum_{i=1}^{\infty}\left[(1-p_o)(p_o^{i-1} - p_s(i,w))u_o^i \cdot\frac{1}{m}\sum_{j=0}^{m-1} p_v((i+j)/m,k)\right]. \tag{22}$$

ACKNOWLEDGMENTS

The authors would like to thank Tin-Fook Ngai, Qinglong Hu, and the anonymous reviewers for their valuable comments and suggestions that improved the quality of this paper. The research was supported by the Research Grant Council, Hong Kong SAR, China, under grant numbers HKUST-6077/97E and HKUST-6241/00E.

REFERENCES

[1] A. Acharya, B.R. Badrinath, T. Imielinski, and J.C. Navas, "A WWW-Based Location-Dependent Information Service for Mobile Clients," technical report, Dept. of Computer Science, Rutgers Univ., July 1995.

[2] I.F. Akyildiz, Y.-B. Lin, W.-R. Lai, and R.-J. Chen, "A New Random Walk Model for PCS Networks," *IEEE J. Selected Areas in Comm. (JSAC),* vol. 18, no. 7, pp. 1254-1260, July 2000.

[3] R. Alonso, D. Barbara, and H. Garcia-Molina, "Data Caching Issues in an Information Retrieval," *ACM Trans. Database Systems (TODS),* vol. 15, no. 3, pp. 359-384, Sept. 1990.

[4] D. Barbara and T. Imielinksi, "Sleepers and Workaholics: Caching Strategies for Mobile Environments," *Proc. ACM SIGMOD Conf. Management of Data,* pp. 1-12, May 1994.

[5] O.A. Bukhres and J. Jing, "Performance Analysis of Adaptive Caching Algorithms in Mobile Environments," *Information Sciences,* vol. 95, no. 1-2, pp. 1-27, Nov. 1996.

[6] J. Cai and K.-L. Tan, "Energy-Efficient Selective Cache Invalidation," *ACM/Baltzer J. Wireless Networks (WINET),* vol. 5, no. 6, pp. 489-502, 1999.

[7] J. Cai, K.-L. Tan, and B.C. Ooi, "On Incremental Cache Coherency Schemes in Mobile Computing Environments," *Proc. 13th Int'l Conf. Data Eng. (ICDE '97),* pp. 114-123, Oct. 1997.

[8] G. Cao, "A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments," *Proc. Sixth Ann. ACM/IEEE Int'l Conf. Mobile Computing and Networking (MobiCom '2000),* pp. 200-209, Aug. 2000.

[9] B.Y.L. Chan, A. Si, and H.V. Leong, "Cache Management for Mobile Databases: Design and Evaluation," *Proc. 14th Int'l Conf. Data Eng. (ICDE '98),* pp. 54-63, Feb. 1998.

[10] K. Cheverst, N. Davies, K. Mitchell, and A. Friday, "Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE Project," *Proc. Sixth Ann. ACM/IEEE Int'l Conf. Mobile Computing and Networking (MobiCom '2000),* pp. 20-31, Aug. 2000.

[11] S. Dar, M.J. Franklin, B.T. Jonsson, D. Srivatava, and M. Tan, "Semantic Data Caching and Replacement," *Proc. 22nd Int'l Conf. Very Large Data Bases (VLDB '96),* pp. 330-341, Sept. 1996.

[12] M.H. Dunham and V. Kumar, "Location Dependent Data and Its Management in Mobile Databases," *Proc. Ninth Int'l Workshop Database and Expert Systems Applications,* pp. 414-419, Aug. 1998.

[13] C.C.F. Fong, J.C.S. Lui, and M.H. Wong, "Quantifying Complexity and Performance Gains of Distributed Caching in a Wireless Network Environment," *Proc. 13th Int'l Conf. Data Eng. (ICDE '97),* pp. 104-113, Oct. 1997.

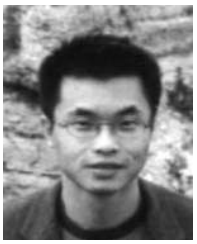
[14] L. Forlizzi, R.H. Güting, E. Nardelli, and M. Schneider, "A Data Model and Data Structures for Moving Objects Databases," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* pp. 319-330, May 2000.

[15] I.A. Getting, "The Global Positioning System," *IEEE Spectrum,* vol. 12, no. 30, pp. 36-38, 43-47, Dec. 1993.

[16] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster, "The Anatomy of a Context-Aware Application," *Proc. Fifth Ann. ACM/IEEE Int'l Conf. Mobile Computing and Networking (MobiCom '99),* pp. 59-68, Aug. 1999.

[17] Q.L. Hu and D.L. Lee, "Cache Algorithms Based on Adaptive Invalidation Reports for Mobile Environments," *Cluster Computing,* vol. 1, no. 1, pp. 39-48, Feb. 1998.

[18] Q.L. Hu, D.L. Lee, and W.-C. Lee, "Performance Evaluation of a Wireless Hierarchical Data Dissemination System," *Proc. Fifth Ann. ACM/IEEE Int'l Conf. Mobile Computing and Networking (MobiCom '99),* pp. 163-173, Aug. 1999.
[19] J. Jing, A.K. Elmagarmid, A. Helal, and R. Alonso, "Bit-Sequences: A New Cache Invalidation Method in Mobile Environments," *Baltzer J. Mobile Networks and Applications (MONET),* vol. 2, no. 2, pp. 115-127, 1997.
[20] D. Lam, D.C. Cox, and J. Widom, "Teletraffic Modeling for Personal Communications Services," *IEEE Comm. Magazine,* vol. 35, no. 2, pp. 79-87, Feb. 1997.
[21] V. Persone, V. Grassi, and A. Morlupi, "Modeling and Evaluation of Prefetching Policies for Context-Aware Information Services," *Proc. Fourth Ann. ACM/IEEE Int'l Conf. Mobile Computing and Networking (MobiCom '98),* pp. 55-65, Oct. 1998.
[22] E. Pitoura and P.K. Chrysanthis, "Exploiting Versions for Handling Updates in Broadcast Disks," *Proc. 25th Int'l Conf. Very Large Data Bases (VLDB '99),* pp. 114-125, Sept. 1999.
[23] N.B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The Cricket Location-Support System," *Proc. Sixth Ann. ACM/IEEE Int'l Conf. Mobile Computing and Networking (MobiCom '2000),* pp. 32-43, Aug. 2000.
[24] Q. Ren and M.H. Dunham, "Using Clustering for Effective Management of a Semantic Cache in Mobile Computing," *Proc. Int'l Workshop Data Eng. for Wireless and Mobile Access (MobiDE '99),* pp. 94-101, Aug. 1999.
[25] Q. Ren and M.H. Dunham, "Using Semantic Caching to Manage Location Dependent Data in Mobile Computing," *Proc. Sixth Ann. ACM/IEEE Int'l Conf. Mobile Computing and Networking (MobiCom '2000),* pp. 210-221, Aug. 2000.
[26] S. Saltenis, C.S. Jensen, S.T. Leutenegger, and M.A. Lopez, "Indexing the Positions of Continuously Moving Objects," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* pp. 331-342, May 2000.
[27] M. Satyanarayanan, J.J. Kistler, P. Kumar, M.E. Okasaki, E.H. Siegel, and D.C. Steere, "Coda: A Highly Available File System for a Distributed Workstation Environment," *IEEE Tran. Computers,* vol. 39, no. 4, pp. 447-459, Apr. 1990.
[28] H. Schwetman, *CSIM User's Guide (Version 18),* MCC Corporation, http://www.mesquite.com, 1998.
[29] S. Shekhar, A. Fetterer, and D.-R. Liu, "Genesis: An Approach to Data Dissemination in Advanced Traveler Information Systems," *IEEE Data Eng. Bull.,* vol. 19, no. 3, pp. 40-47, Sept. 1996.
[30] A.P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao, "Modeling and Querying Moving Objects," *Proc. 13th Int'l Conf. Data Eng. (ICDE '97),* pp. 422-432, Apr. 1997.
[31] K.L. Tan and J. Cai, "Broadcast-Based Group Invalidation: An Energy-Efficient Cache Invalidation Strategy," *Information Sciences,* vol. 100, no. 1-4, pp. 229-254, 1997.
[32] M. Taylor, W. Waung, and M. Banan, *Internetwork Mobility: The CDPD Approach.* N.J.: Prentice Hall, 1997.
[33] J. Wang, "A Survey of Web Caching Schemes for the Internet," *ACM Computer Comm. Rev.,* vol. 5, no. 29, pp. 36-46, Oct. 1999.
[34] O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang, "Moving Objects Databases: Issues and Solutions," *Proc. 10th Int'l Conf. Scientific and Statistical Database Management (SSDBM '98),* pp. 111-122, July 1998.
[35] K.-L. Wu, P.S. Yu, and M.-S. Chen, "Energy-Efficient Caching for Wireless Mobile Computing," *Proc. 12th Int'l Conf. Data Eng. (ICDE '96),* pp. 336-343, Feb. 1996.
[36] J. Xu and D.L. Lee, "Querying Location-Dependent Data in Wireless Cellular Environment," *WAP Forum/W3C Workshop Position Dependent Information Services,* Feb. 2000.
[37] J. Xu, X. Tang, D.L. Lee, and Q.L. Hu, "Cache Coherency in Location-Dependent Information Services for Mobile Environment," *Proc. First Int'l Conf. Mobile Data Access (MDA '99),* pp. 182-193, Dec. 1999.



Jianliang Xu received the BEng degree in computer science and engineering from Zhejiang University, Hangzhou, China in 1998, and the PhD degree in computer science from Hong Kong University of Science and Technology in 2002. He is an assistant professor in the Department of Computer Science at Hong Kong Baptist University. His research interests include mobile computing, wireless networks, Internet technologies, and distributed systems. He is a member of the IEEE.

Xueyan Tang received the BEng degree in computer science and engineering from Shanghai Jiao Tong University, Shanghai, China in 1998. He is currently a PhD candidate in the Department of Computer Science at the Hong Kong University of Science and Technology. His research interests include web caching and performance, Internet technologies, mobile computing, and distributed systems.

Dik Lun Lee received the MS and PhD degrees in computer science from the University of Toronto. Canada in 1981 and 1985, respectively. He joined the Hong Kong University of Science and Technology as an associate professor in 1994, and is now a professor in the Department of Computer Science. Before joining HKUST, he was an associate professor of computer and information science at Ohio State University, Columbus, Ohio. He has served as program committee member and the chair for numerous international conferences, and was the founding conference chair for the International Conference on Mobile Data Access/Management. His research interests include document retrieval and management, discovery, management and integration of information resources on Internet, and mobile and pervasive computing. He was the chairman of the ACM Hong Kong Chapter.