Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Information Systems 29 (2004) 207–234

www.elsevier.com/locate/infosys

Concurrency control strategies for ordered data broadcast in mobile computing systems[☆], [☆☆], [☆☆☆]

Kam-Yiu Lam*, Edward Chan, Hei-Wing Leung, Mei-Wai Au

Department of Computer Science, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon, Hong Kong, PR China

Received 24 October 2001; accepted 11 July 2002

Abstract

Although data broadcast has been shown to be an efficient method for disseminating data items in mobile computing systems, the issue on how to ensure consistency and currency of data items provided to mobile transactions (MT), which are generated by mobile clients, has not been examined adequately. While data items are being broadcast, update transactions may install new values for them. If the executions of update transactions and the broadcast of data items are interleaved without any control, mobile transactions may observe inconsistent data values. The problem will be more complex if the mobile clients maintain some cached data items for their mobile transactions. In this paper, we propose a concurrency control method, called *ordered update first with order* (*OUFO*), for the mobile computing systems where a mobile transaction consists of a sequence of read operations and each MT is associated with a time constraint on its completion time. Besides ensuring data consistency and maximizing currency of data to mobile transactions, OUFO also aims at reducing data access delay of mobile transactions using client caches. A hybrid re-broadcast/invalidation report (IR) mechanism is designed in OUFO for checking the validity of cached data items so as to improve cache consistency and minimize the overhead of transaction restarts due to data conflicts. This is highly important to the performance of the mobile computing systems where the mobile transactions are associated with a deadline constraint on their completion times. Extensive simulation experiments have been performed to compare the performance of OUFO with two other efficient schemes, the multi-version broadcast method and the periodic IR method. The performance results show that OUFO offers better performance in most aspects, even when network disconnection is common.
© 2002 Elsevier Science Ltd. All rights reserved.

Keywords: Concurrency control; Data broadcast; Real-time data and mobile transactions processing

1. Introduction

The research in mobile computing has attracted a lot of attention in recent years. Owing to the intrinsic constraints of mobile computing systems, such as asymmetric bandwidth, limited power supply and unreliable communication, the design

[☆] A preliminary version of the paper appeared in the *Proceedings of 2000 International Conference on Management of Data* (COMAD'2000), Pune, India, December 2000.

[☆☆] The work reported in this paper was supported in part by the City University of Hong Kong under the strategic grants numbers 7001117 and 7001069.

[☆☆☆] Recommended by P. O'Neil.

*Corresponding author. Tel.: +852-2788-9807; fax: +852-2788-8614.

E-mail addresses: cskylam@cityu.edu.hk (K.-Y. Lam); csedchan@cityu.edu.hk (E. Chan).

of an efficient and cost-effective mobile computing system poses many challenges [1–5]. One of the most important issues is how to efficiently disseminate consistent data items from a database server to transactions, called *mobile transactions* (*MT*), which are generated by mobile clients [2,3]. In recent years, various efficient data dissemination methods were proposed [1,6–9]. Most of them are based on data broadcast in which a broadcast server periodically and continuously broadcasts data items to a population of mobile clients through a mobile network. The main advantage of data broadcast is that the broadcast of a data item can meet the data requirements of several mobile transactions, resulting in an efficient use of the limited mobile bandwidth. Under data broadcast, mobile transactions do not need to inform the broadcast server before accessing a data item. They can get the data item from the "air" while it is being broadcast.

Since many data items in a mobile computing system are used to record the *real-time information* in the system, e.g., the current traffic conditions of the roads, the last traded prices of stocks, and news updates, their values could be highly dynamic [10]. Updates, which are generated from external devices, capture the most current information for the system and refresh the values of the data items [11,12]. Allowing execution of updates to be interleaved with data broadcast is important in maintaining the "freshness" of the data items [11]. Accessing outdated (stale) data items is undesirable and may significantly affect the usefulness of the information to mobile clients [2,12,13]. However, if concurrent execution of updates and data broadcast is allowed, the problem of concurrency control must be addressed. Otherwise, the resulting execution schedule between the mobile and update transactions may be non-serializable [14] and consistency of data items provided to mobile transactions cannot be guaranteed.

The result of providing inconsistent data items to a mobile client could be very serious. Consider the case that a mobile client may ask the exchange rates of US dollars to UK pounds in New York and London at the same time. If the returned results are inconsistent, i.e., they are representing the information at different time points, the mobile client may make a wrong trading decision. Similarly, if the information of the last traded price of a stock is not consistent with the stock index, i.e., there is a rise in the last traded price of the stock while the stock index drops, the mobile client does not know which one is correct and neither piece of information will be useful.

Unfortunately, conventional concurrency control protocols, such as two-phase locking, optimistic concurrency control method and timestamp ordering [14], are not suitable to mobile computing systems as the overheads for setting locks and detecting data conflicts in a mobile environment could be very expensive [15]. Therefore, new concurrency control methods, which take into considerations the data properties and the characteristics of the systems, are needed.

In this paper, we study the problem of broadcasting consistent data items to *read-only mobile transactions* while allowing updates to be performed concurrently at the broadcast server. It is assumed that each mobile transaction is associated with a deadline on its completion time. Although it is not a hard real-time deadline, it is important to meet the deadline. Otherwise, the usefulness of completing a transaction will be greatly affected.

The problem of providing consistent data items for executions of mobile transactions has received growing interest in recent years [4,15–17]. An efficient and pioneering method is by broadcasting multiple versions of data items [18]. Consistent data items are provided to mobile transactions by requiring mobile transactions to read data items committed at the same point of time, i.e. the last broadcast cycle. The basic multi-version (MV) broadcast method is extended in [17,19] for systems with client caches where multiple versions of data items are maintained. By reading cached data items, which are committed at the same time, the data access delay can be much reduced and at the same time data consistency is maintained.

Another efficient method for concurrency control between read-only mobile transactions and update transactions is a data re-broadcast scheme called *update first with ordering* (*UFO*) [16]. Although UFO can provide the most updated values of data items to mobile transactions (high currency) and at the same time maintain the serializability of execution between update and mobile transactions, it is designed for mobile transactions with unordered operations, i.e., the operations in a transaction can be performed in any order. In this paper, we extend the UFO method to *ordered UFO* (*OUFO*) for mobile transactions whose data requirements are ordered. In addition, we have designed an invalidation report (IR) scheme in OUFO to ensure data consistency of cached data items and at the same time reduce the restart overheads of transactions as a result of data re-broadcast. (The second problem is a serious performance problem in the original UFO.) We have performed extensive simulation experiments to compare the performance of OUFO with the MV broadcast method [17,19] and the periodic invalidation method (IR) where IRs are broadcast from the broadcast server to mobile clients to validate the accessed data items of the mobile transactions [17]. The following summarizes the main contributions of the paper:

(1) OUFO, which is an extension of the original UFO, is proposed for mobile transactions whose read operations are ordered. In the design of the algorithms, we aim to meet transaction deadlines, providing consistent data items to mobile transactions and maximizing data currency to the transactions.
(2) A hybrid re-broadcast/invalidation scheme is designed in OUFO to ensure data consistency at the client caches and at the same time minimize the cost of transaction restart due to data conflict. The effect of network disconnection on OUFO is investigated and its performance under a frequent disconnection environment is studied.
(3) Extensive simulation experiments are performed to investigate how well our scheme ensures the currency of data items and at the same time maintains serializability of transaction execution. Its performance is compared with two other efficient schemes, the MV broadcast method (MV), and the periodic invalidation method (IR), under various system settings.

The organization of the remaining parts of the paper is as follows. Section 2 reviews related work on broadcasting consistent data items. Section 3 introduces the system model and the assumptions of the model. Section 4 introduces the OUFO method together with a detailed discussion on its correctness, properties and implementation overhead. The problem of network disconnection on OUFO is also discussed. Section 5 is a performance study in which we compare the performance of OUFO with the MV broadcast with MV cache method and the periodic invalidation method. We conclude the paper in Section 6.

2. Related work

Providing consistent data items to transactions is one of the most important requirements of a transaction processing system. However, traditional concurrency control protocols are not suitable for mobile computing systems due to their heavy overheads in detecting data conflicts in a mobile environment [4,11]. To our knowledge, few studies until now have been done in this area although it is an important issue. One suggested solution is to relax the consistency requirement. In [4], a two-level consistency model is proposed. Semantically related data are grouped together into a cluster, and the data items inside a cluster are mutually consistent. A certain degree of inconsistency is allowed among data items at different clusters.

In [15], a control matrix-based scheme is proposed for data conflict detection and resolution. For a database of n data items, a matrix of size $n \times n$ is used. In each broadcast cycle, the control matrix is broadcast together with the data items. A mobile client performs consistency checking using the matrix before reading any data item from the “air”. This method can handle read-only transactions as well as update transactions. The write operations are performed on local copies of the data items at the client. At the end of a transaction, the whole transaction including all of the read and write operations and the cycle numbers in which they are performed will be sent to the server for commitment.

Another method to detect the non-serializability problem is to broadcast serialization graphs [18]. Each client maintains its local serialization graph to ensure that the schedule of its committed transactions is serializable with respect to the update transactions at the broadcast server. The main drawback of this method is the heavy overhead in broadcasting the serialization graphs since every data conflict at the database server has to be broadcast. Furthermore, each client must listen to the transmission channel continuously to maintain and update its serialization graph. This leads to a serious problem when disconnection occurs between a mobile client and the broadcast server. The mobile client cannot obtain the updated serialization information about its transactions, making it virtually impossible to ensure the serializability of transaction execution when network disconnection occurs.

A method similar to the broadcast serialization graph method is the IR method [17] in which an IR is periodically broadcast before each broadcast cycle. The IR consists of a list of identities of data items, which have been updated at the broadcast server within the previous broadcast cycle. The validity of the data items accessed by a mobile transaction is ensured by checking against the IR. A transaction has to be restarted in case any of its accessed data items is invalid. Although the periodic IR method can maximize the currency of the data items provided to mobile transactions, a large number of transactions will be restarted if the update rate of the data items is high.

In [17,19], a MV broadcast method is proposed in which the server broadcasts previous versions of data items in addition to the committed versions of the data items at the last broadcast cycle. If a mobile transaction wants to access a data item, it will get the latest version for its first read operation. The subsequent read operations of the transaction will read the data items with the largest version number which is smaller than or equal to the data version of the first operation. By allowing a transaction to read an older version of a data item, data consistency can be ensured at the expense of currency, i.e., a mobile transaction may not receive the latest value of a data item. In order to reduce the number of versions to be broadcast and to facilitate the checking of consistency, update transactions will update the database only at the end of a broadcast cycle even though they arrive in the middle of a broadcast cycle. The number of versions to be broadcast for a data item is determined by the life span of the transactions. It is defined as the maximum number of broadcast cycles from which a transaction may read data items. Another important assumption of the MV broadcast method is that at least one value (the current one) is broadcast for each data item in each cycle.

The MV broadcast method is subsequently extended for systems with client caches [17,19]. In addition to broadcasting multiple versions of a data item, the clients also maintain previous versions of data items at their caches. The number of versions to be maintained at a client cache is again determined by the life spans of the transactions. The same rule for accessing broadcast data is used for accessing cached items. The efficiency and characteristics of the MV methods as compared with other methods, such as the serialization graph broadcast, have been examined in [17]. The MV method is very useful for systems where the mobile clients are frequently disconnected from the mobile network since the mobile clients may access the cached data items while they are disconnected.

The problem of broadcasting real-time data items to time-constrained transactions also has received growing research interest in recent years. In [15], a hybrid broadcast on-demand (BoD) model is presented as an extension of the traditional broadcast disk model [20] for systems where the data items have real-time periodic properties. In the BoD model, the design of a broadcast schedule is based on the update periods of the data items. In [21,22], an adaptive data dissemination model and a time-critical adaptive hybrid data dissemination (TC-AHB) method were proposed. In the proposed method, both on-demand and push are employed in the design of a broadcast schedule and up-link channels are used to collect the on-demand data requests from mobile transactions.

3. System model

The mobile computing system model consists of a broadcast server, a number of mobile clients, a database, and a mobile network as shown in Fig. 1. The broadcast server communicates with the mobile clients through low bandwidth wireless channels of the mobile network. The broadcast server maintains a database. It is assumed that the values of the data items are highly dynamic since they record real-time information in the system, e.g., last traded price of the stocks and the locations of moving objects. The target applications are mobile information systems, which require the retrieval of real-time information for decision-making, i.e., mobile stock information systems, navigation systems and location management of moving objects. In such kind of systems, reading outdated data items will significantly affect the usefulness of the information. For example, in a navigation system, a driver may generate a navigation request, which requires access to the real-time traffic conditions of the roads connecting its current position to its destination. The result of the query is important to his decision in choosing the best path to his destination. If the query accesses outdated data, the probability of arriving late at the destination will be increased.

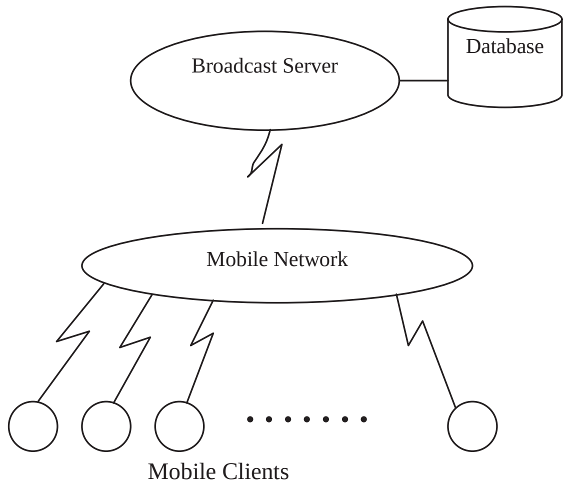

Fig. 1. System model.

To maintain the validity of the data items, update transactions are generated to refresh the data values. It is assumed that the update transactions are generated from some external devices, or provided from a data vendor, i.e., Reuters. Each update transaction is associated with a time stamp, which is its generation time. It is used to indicate at which snapshot of the external environment the update is generated. The time stamp is recorded together with the new value into the data item as its version number. In the model, it is assumed that update transactions are short transactions, consisting of one to several write operations, e.g., stock quotes and news updates, with a high arrival rate. It is further assumed that a well-formed concurrency control protocol, such as two-phase locking [14], is used for concurrency control amongst the update transactions at the broadcast server.

The broadcast server periodically broadcasts data items one by one, continuously, until the end of a broadcast cycle. Then, the next broadcast cycle starts immediately. The length of a broadcast cycle may be fixed or variable depending on the adopted broadcast scheduling algorithm, which selects data items for broadcast. In the last few years, many efficient broadcast algorithms have been proposed based on the deadlines of transactions or the access frequencies of data items [23,8,15]. In this paper, we assume the use of a simple flat disk for selecting data items for broadcast. Note that the proposed OUFO method can be applied to many different broadcast-scheduling algorithms without any changes and its impacts on the broadcast algorithms are minimal in most cases.

At each mobile client, some recently accessed data items are maintained at the client cache. Since the cache size of most mobile clients is very limited, it is assumed that only a small number of data items can be maintained at the client cache. Caching data items at mobile clients is important to the execution of mobile transactions. If a mobile transaction can find its required data items at the client cache, it does not need to wait for them from the broadcast cycle. This can greatly reduce the data access delay. It is assumed that the least recently used method is used for cached data replacement for its simplicity and popularity.

Mobile clients issue MTs to access data items sporadically. It is assumed that each MT consists of a sequence of read data requests (read operations). To simplify the discussion, each operation requires access to one data item. The operations in an MT are ordered and pre-defined with precedence relationships. The execution of the operations of an MT has to follow the pre-defined precedence relationships. In processing an operation, the system first searches the cache of its initiating client for the required data item of the operation. If the data item cannot be found at the cache, the MT will wait for it from the broadcast cycle. After getting the data item, the next operation of the MT may start. The results of a mobile transaction will be reported to the requesting mobile client only at its commit time. Thus, if an MT has to be aborted or restarted from beginning, the atomicity property can be maintained by completing an undo operation for the transaction.

Each MT is defined with a deadline. The period between the arrival time of a transaction and its deadline is called its life span. It is assumed that the deadlines of the mobile transactions are firm. A transaction will become totally useless after its deadline and has to be aborted. It is assumed that the set of transactions in the system has the same life span. It is an application-dependent parameter. It indicates how long a transaction is allowed to exist in the system. It may be determined based on the dynamic properties of the data items to be accessed by the transactions. For example in stock trading systems, due to the highly dynamic properties of stock data, the transactions may be defined with tight deadlines.

In executing an MT, both data consistency and currency are important. In this paper, we adopt serializability as the correctness criterion for database consistency [14] as it has been commonly used in conventional database systems and widely accepted in the database community. If the serialization graph of a set of transactions is acyclic, then the schedule is serializable. Another important performance requirement is to maximize data currency provided to mobile transactions. In a mobile computing system, while data items are being broadcast from the database server to mobile clients, updates may arrive and new versions may be installed into the database. When an MT reads a data item from the broadcast channel, the version may not be the latest one. In this paper, we define currency based on the stale access rate, which is the percentage of outdated data versions observed by the transactions. A high stale access rate means low currency, i.e., more outdated information is being observed.

In our model, the broadcast process is modeled as a long read-only transaction, called BT. The length of a BT is defined based on the life span of the mobile transactions such that the time required to broadcast all its data items is equal to the life span of a mobile transaction. Thus, the data item set of a BT includes all the data items, which are broadcast during the period from (current time−life span of a mobile transaction) to current time. Note that the concept of BT is logical only and a BT actually does not possess any characteristics required for a traditional transaction, e.g., ACID properties. The reason for treating it as a transaction is to facilitate the discussion of the mechanism and correctness of the OUFO, which will be introduced in Section 4. Note that according to the above definition, each MT will have its BT for reading its required data items and no MT will read data items from more than one BT since the length of its BT is defined according to its life span.

The assumptions of the system model are summarized below:

(1) All update transactions are short transactions and are processed at the broadcast server.
(2) The arrival rate of update transactions is high and sporadic.
(3) All mobile transactions are *read-only* and the read operations in a MT are *ordered*.
(4) Each mobile transaction has a firm deadline, i.e., arrival time + life span. It is important to complete a mobile transaction before its *deadline*. Otherwise, it is useless and has to be aborted.
(5) The result of a mobile transaction is reported to the requesting client at its commit time.
(6) Each mobile client maintains some recently accessed data items at its cache. It is assumed that the least recently used method is used for cached data replacement.
(7) The processing requirement of a mobile transaction ensures that all the data items accessed by an MT is consistent and at the same time the data currency is maximized.
(8) The broadcast process is modeled as a BT and the length of a BT is defined from the life spans of the MTs. Each MT will have its own BT for reading its required data items.

4. Ordered update first with order (OUFO)

4.1. Method overview

In [16], the UFO method is proposed for broadcasting consistent data items to mobile transactions with unordered read operations, i.e., each mobile transaction consists of a set of read operations and the operations can be executed in any order. In this paper, we propose an extension of UFO, which we call the OUFO for mobile transactions in which the read operations are ordered such that the read operations in a transaction have precedence relationships and their execution has to satisfy the defined precedence constraints, i.e., they must be executed one after another. For example, the second operation in a transaction can only start after the completion of the first operation of the transaction. The issues of accessing consistent cached data items and processing operations under disconnection are also addressed.

The basic principle used in OUFO to ensure data consistency is that if a data conflict occurs between a BT and an update transaction (U), the serialization order between them will always be $U \rightarrow BT$. Since MTs read data items from BTs, the serialization order between update transactions and mobile transactions will always be $U \rightarrow MT$. Thus the schedules will always be serializable. (The correctness of the OUFO method will be discussed in more detail in Section 4.5.1.)

Basically, the OUFO method consists of three parts:

(1) execution of update transactions;
(2) conflict resolution between update and BTs; and
(3) consistency check for cached data items.

4.2. Execution of update transactions

The execution of update transactions in OUFO is the same as in the original UFO. It is divided into two phases: the *execution phase* and the *update phase*. During the execution phase, the operations of an update transaction are executed and data conflicts with other update transactions are resolved using a conventional concurrency control protocol such as 2PL. The new values from the write operations of a transaction are written in a private workspace of the transaction instead of into the database immediately. When all the operations of an update transaction have been completed, it enters the update phase in which permanent updates of the database will be performed by copying the new values from its private workspace into the database. Data conflict with the BT will be checked in the update phase, which is performed in a critical section. So, we can see that the update transactions adopt 2PL for resolving data conflicts with other update transactions and use an optimistic approach [14] to detect conflicts with the BT.

There are two important advantages in dividing the execution of the update transactions into two phases. Firstly, it can significantly reduce the blocking probability and blocking delay of a BT. If we do not divide its execution into two phases and a BT wants to read a data item, which is already locked by an update transaction, the BT will be blocked until the update transaction is committed according to the principles of 2PL [14]. At the same time, the update transaction, which is holding the lock, may be blocked due to data conflicts with other update transactions. Due to transitive blocking, the blocking time of a BT can be very long. Dividing the execution of an update transaction into two phases can greatly reduce the blocking probability and blocking time of BTs since data conflicts between update and BTs will occur only when the update transaction is in the update phase, which is much shorter. Secondly, the detection of data conflicts between the update and BTs will become much easier. At the update phase, the system knows which data items have been accessed by the update transaction. By comparing the write set of the update transaction with the read set of the BT, the system can easily determine whether there is any data conflict between them. As we see below, such new updates during a BT will be re-broadcast to the mobile clients.

4.3. *Conflict resolution between update and BTs*

Data conflict between an update transaction and a BT will be detected when the update transaction enters its update phase in which a *re-broadcast scheme* is used to resolve the conflict. The purpose is to reverse the serialization order from BT→U to our desired order, U→BT. The details of the algorithms at both the broadcast server and mobile clients are shown in the following sub-sections.

4.3.1. *Algorithm at the broadcast server*

The following defines the algorithm at the broadcast server. It is performed when an update transaction enters the update phase.

```
$O_{BT}$ = set of data items of the broadcast transaction, BT
$O_U$ = set of data items of the update transaction, U
if  $O_{BT} \cap O_U = \{\}$
    BT and U have no dependency
else
    for each data item $d_i \in \{O_{BT} \cap O_U\}$
            re-broadcast data item $d_i$
endif
```

Note that at the update phase, the set of data items to be updated by an update transaction is already defined. By re-broadcasting the conflicting data item, the serialization order between the BT and the update transaction is reversed. As described in Section 3, the set of data items in a BT consists of those data items which are broadcast in the period from (current time−life span of a mobile transaction) to current time. Thus, the set of data items in the BT is not fixed. After the broadcast of a data item, the data item will be included in the BT and the first data item of the BT will be removed. Then, the checking of data conflict with update transactions will be performed according to the above algorithm. A data item will be re-broadcast if they exist in both update and BTs.

4.3.2. *Algorithm at the mobile client*

The data items requested by an MT are represented by a sequence of read operations. Processing of an MT starts from the first read operation in the sequence. Each data item received from a BT is matched with the requesting data item of the executing operation. If there is a match, the MT will read the data item and the operation will be processed. The matching process is repeated for the next read operation until there is no more operation in the sequence, and then the transaction will commit. In case a data item, which has been read by the MT, is re-broadcast while the MT is waiting for other data items, the MT will be restarted from the operation which requests that data item. It will use the re-broadcast value for the execution of the operation. There are two reasons for the restart. Firstly, it is to ensure the desired serializability order U → MT. The second reason is to provide the most updated data values to the mobile transactions. The algorithm at the mobile client is shown below where it is assumed that the MT reads all its data items from BT and there is no cache at the client. We will discuss the case of accessing cached data items in the next subsection.

```
$L_{MT}$ = the sequence of read operations in MT
$S_{MT}$ = the set of data items have been accessed by MT
$d_c$ = the data item required by the first operation in $L_{MT}$
$S_{MT}$ = {}
loop until $L_{MT}$ exhausted
    read data item $d_i$ from BT ($d_i$ is the currently broadcasting item)
    if    $d_i = d_c$
    then  MT processes $d_i$ and updates $L_{MT}$
    else
        if   $d_i \in S_{MT}$
            MT repeats the processing on $d_i$ , updates $L_{MT}$
            Restarts its execution from the operation which requires $d_i$.
        endif
    endif
    $d_c$ = the data item requested by the next operation in $L_{MT}$
end loop
```

Note that in order to maintain the order of executions of the operations, if an operation has to be aborted, the operations depending on the aborted operation in execution order also have to be aborted. A mobile transaction may be restarted several times if its accessed data items are updated several times before its completion. However, this is the cost we need to pay for ensuring the correctness of the results and the ACID properties of the transactions. To minimize the impact and costs, OUFO maintains cached data so that a restarted transaction only needs a short time to return to its restart point.

4.4. Consistency checking for cached data items

4.4.1. Benefit of caching data items

If a mobile transaction has to read all its requested data items directly from a BT, the mobile transaction may need to wait for a long time to get all its required data items. The waiting time depends on the broadcast algorithm adopted and the size of the database if a flat broadcast disk is used [20]. Another potential performance problem of OUFO is that a mobile transaction has to be restarted if any one of its previously accessed data items is broadcast again before the transaction commits. Restarting a mobile transaction may greatly increase its response time since it has to wait for its required data items from the broadcast cycle again. Even worse, it may be rescheduled repeatedly until it misses its deadline.

To reduce the data access delay and reduce restart overheads, a mobile client may maintain the recently accessed data items at its cache. In case a mobile transaction is restarted due to the arrival of a new version of one of its accessed data items, the restarted transaction can immediately access its previously accessed data items from the cache instead of waiting for them to appear in the BT. In OUFO, an invalidation mechanism (to be described in the next sub-section) is designed to ensure that all the cached data items accessed by a committed transaction are consistent. In addition, after a mobile transaction has accessed a data item, the data item will also be placed in the client cache. If the cache is full, a cache replacement algorithm, such as the Least Recently Used method, will be invoked. It is assumed that an *auto-refreshment scheme* is adopted to maintain the validity of the cached data items. Whenever a data item is being broadcast and the cache has a copy of the data item, the cached copy will be refreshed with the version currently broadcast.

4.4.2. IR-based consistency checking

Unlike getting data items directly from a BT, where the data items are guaranteed to be the most updated and consistent version, cached data items can be outdated and inconsistent. The probability of finding an invalid version at the cache depends on how the system selects data items for broadcast and the update rates of the data items. To check the validity of the cached data items, an IR will be generated periodically and interleaved with the broadcast of data items.

4.4.2.1. Generation of IR. An IR is prepared and periodically broadcast by the server. The period for generating the IR is called *report period.* An IR includes the latest update time stamps and identities of the set of data items, which are updated during the interval from (current time− report duration) to current time, i.e., the updates occurred during the last report duration. The report duration is the time frame over which a report collects the update information of the data items. It is assumed that a data item, whose update time is beyond the report duration, will be too "old" to be useful. Each IR will be shifted by a report period from the last report generation time. The report duration is chosen to be much greater than the report period, i.e., a multiple of a report period. Using such a sliding window approach for generating the reports can help to resolve the problem of validation under disconnection. A mobile transaction can still validate its accessed data items if it has disconnected from the network for a period not longer than the report duration since the identity of an updated data item will be repeated in several IRs for a period equal to the report duration. Note that even though a large value is used for the report duration, the size of a report may not be very large. Its size depends on the number of updates occurred during the report period. In the worst case, a report will include all the data items, which are updated in the report duration.

For example in Fig. 2, data items x, y and z are updated at time t_3, t_4 and t_5, respectively. Then, data items x and z are updated at t_6 and t_7, respectively. Report R1 is generated at time t_8 and it covers the update information for the update period from t_8 to t_8-report duration. The identity and latest time stamps of x, y and z will be included in R1. If report R2 is generated t_8 plus the report period, it will include the latest time stamps of data items x and z.

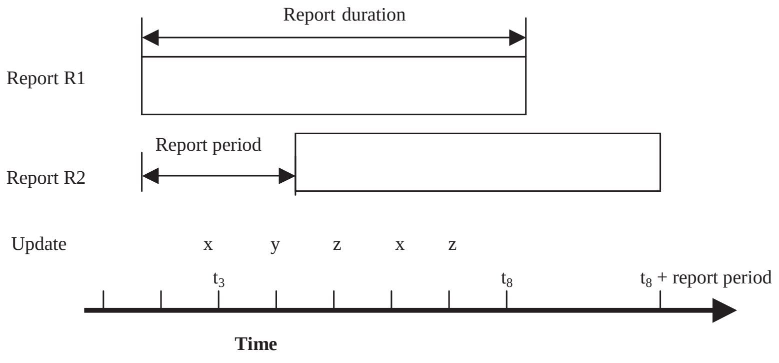

Fig. 2. Generation of the IR.

4.4.2.2. Validation. It is assumed that whenever a mobile client puts a broadcast data item into the cache, it also records down the broadcast time of the data item. Based on the broadcast time, the cached data items are divided into two groups: *newest version* and *unknown version*. A data item belongs to the newest version group if its broadcast time + length of the BT > current time. Otherwise, they are classified into the unknown version group.

When a mobile transaction wants to access a data item, the transaction can access the data item immediately if it can be found at the cache. Otherwise, it has to wait for the data item from the BT. If a mobile transaction has completed all its operations, it will check the validity of its accessed data items. If all of them belong to the newest version group, the transaction can commit immediately. Otherwise, the mobile transaction cannot commit until it receives an IR and has validated its accessed data items.

Validation is done by comparing the time stamps of the data items accessed by the transaction from the cache with the time stamps of the data items in the IR. A data item in the cache is invalid if it is found that the time stamp of the same data item in the validation report is greater, i.e., a new version has been created at the database server. For this case, the mobile transaction has to be restarted from the operation, which has read the invalid data item. Note that if we want to increase the number of data items belonging to the newest version group so as to reduce the blocking delay of mobile transactions for the IR, we can increase the length of the BT. Essentially, it means using a larger set of data items in the BT for checking of data re-broadcast with update transactions. Of course, the tradeoff is between smaller numbers of mobile transactions needed to validate against the IR, and a larger number of re-broadcasts. The decision can be based on the total re-broadcast overhead. If it is low, a longer BT may be used.

A practical problem in implementing the periodic IR is the definition of report period. This parameter actually represents a tradeoff between bandwidth used for broadcasting the IR (which is a valuable resource) and the waiting time of transactions for the report. Since the report size is small and is not expected to consume a large amount of bandwidth, the period for broadcasting the report can be set to a small value in order to reduce the waiting time for validation.

4.4.3. Data re-broadcast vs. IR

Astute readers may notice that OUFO combines two mechanisms to ensure data consistency for mobile transactions:

(1) data re-broadcast basically provides invalidation and immediate data refreshment, and
(2) invalidation by an explicit IR.

The main overhead of the first mechanism is higher broadcast overhead as a result of data re-broadcast. However, its main advantage is smaller waiting time of a mobile transaction to get valid data items in case any of its accessed data items are invalid. The benefit of the second mechanism, explicit IR, is the low broadcast overhead. However, its problem is heavy restart overhead as a transaction has to wait for valid data items again once any of its accessed data item is invalidated. Fortunately, the restart overhead can be much reduced by caching recently accessed data items at the client cache when it is combined with data re-broadcast. IRs determine the validity of a cached data item while data re-broadcast maintains the validity of a cached data item.

In OUFO, the first mechanism is applied to recently broadcast data items while those data items, which are broadcast a long time ago, are validated by IRs. By carefully tuning the time duration of when to apply the first mechanism and when to apply the second mechanism, we can adjust the total overhead of the OUFO method within an acceptable level and at the same time achieve a better performance.

Note that although data re-broadcast may introduce a heavy broadcast overhead when the data conflict probability between update and BT is high, its impact on the overall system performance is not always negative. Many of the proposed broadcast algorithms select data items for broadcast based on the data requirements of the mobile transactions, i.e., data popularity [7,8]. "Hot" data items, i.e., those commonly required by the mobile transactions, will have a higher probability to be selected than the "cold" items in defining data broadcast schedule with the purpose of reducing the waiting time of transactions for hot data items. In data re-broadcast, hot data items may have a higher probability to be re-broadcast since they have a higher probability to be included in a BT (broadcast schedule). Re-broadcasting the previously broadcast data items, which are more likely to be hot data items, may at the same time satisfy the data requirements of other mobile transactions. Consequently, the mean waiting time for data items from the BT may be much reduced.

4.5. *Correctness and implementation overhead*

In this section, we prove the correctness of OUFO based on the serializability theorem [14], which has been widely used in various kinds of database systems. We will show that all the histories or schedules generated from OUFO are serializable.

4.5.1. *Correctness*

Theorem. *The schedules of the committed transactions produced from OUFO method are serializable.*

Proof. The proof consists of two parts. Firstly, we will show that the serialization orders between all committed mobile transactions and their conflicting update transactions will always be $U \rightarrow MT$, if the mobile transactions read the conflicting data items from the BT. In the second part, we will prove that the serialization orders between mobile transactions and update transactions will always be $U \rightarrow MT$, if the mobile transactions read the conflicting data items from the cache.

Let MT, BT and U be mobile transaction, BT and update transaction, respectively. If there is a data conflict between BT and U, and the broadcast of the conflicting data item is before the update of the data item, the serialization between BT and U will be $BT \rightarrow U$. According to OUFO, the conflicting data item will be re-broadcast immediately and the serialization order will be reset to $U \rightarrow BT$. If the broadcast is after the update, the serialization order will be $U \rightarrow BT$. Since MT reads data items from BT, their serialization order will always be $U \rightarrow MT$ for any data conflict between MT and U and the conflicting data item is broadcast during the length of the BT.

If MT reads a data item, which is at the cache or if the data item is broadcast prior to BT, the serialization order between MT and U may be $MT \rightarrow U$. If $MT \rightarrow U$, MT will be restarted after checking with the IR. Thus, the serialization order between MT and U will always be $U \rightarrow MT$. For a non-serializable schedule, the serialization graph must be cyclic, e.g., there must be an edge such that $MT \rightarrow U$. This contradicts the OUFO. □

It is obvious that the currency of the data items observed by mobile transactions can be maximized due to the data re-broadcast and invalidation mechanisms of OUFO since all the data items which are broadcast within the length of a BT are the most updated version and a mobile transaction is not allowed to access outdated data items.

4.5.2. *Implementation overheads*

In this section, we discuss the implementation overheads of OUFO. The main overheads of OUFO at the broadcast server are:

(1) division of the execution of update transactions into two phases;
(2) checking of the data sets of a BT and an update transaction whenever an update transaction wants to enter the update phase;
(3) generation of the IR; and
(4) re-broadcast of conflicting data items if these items are broadcast before the start of the update phase of the update transaction.

Dividing the execution of update transactions into two phases is trivial and should not incur much additional overhead. (This is similar to the deferred update approach [14].) The overhead for checking conflicting data sets and the probability of data conflict should be low as the number of data items to be updated by an update transaction is usually small. To speed up the checking process, the data items to be updated may be sorted according to their IDs.

The main overhead of the method is data re-broadcast. The number of re-broadcast depends on the probability of data conflict between a BT and the update transactions. This in turn depends on the broadcast schedule, arrival rate and the update pattern of the update transactions. Note that as explained at the end of Section 4.4.3, the impact of data re-broadcast is not always negative. It can also help to reduce the waiting time for hot data items at the same time.

The algorithm at the mobile clients is simple. The only additional work is to replace the old version of a data item in case it is re-broadcast, record down the broadcast time, and check with the IR in case a transaction has accessed some data items which belong to the unknown version group.

4.6. Operation under disconnection

An important property of mobile networks is frequent disconnection, which may be voluntary or involuntary. In voluntary disconnection, the mobile client initiates a disconnection from the server in order to conserve power of the mobile machine. Voluntary disconnection of a mobile client will only occur after the completion of its transaction. Thus, it neither affects the mechanism of OUFO nor creates any problem regarding the correctness of the OUFO method. Involuntary disconnection results from instability in mobile network. For example, in a cellular radio network, the strength of signal received by a mobile client is affected by a number of factors such as the distance between the mobile clients and the base station, as well as the height of the surrounding buildings. Disconnection may occur once the signal received by the mobile client is lower than a threshold level. Although the disconnection is usually temporary, its impact on the consistency of transaction execution can be serious.

The main impact of network disconnection on OUFO is that at the time of re-broadcast a mobile client may be disconnected from the network. For this case, the mobile client cannot get the new version of the data item and the serializability order between a mobile transaction and an update transaction cannot be reversed. If they have any further data conflicts, the resulting schedule may be non-serializable. Therefore, to resolve the problem, once a mobile client has been temporarily disconnected from the network, its mobile transaction may consider all its accessed data items to belong to the unknown version group and have to be validated with the IR. Of course, if the disconnection is very long, i.e., greater than the report window size, a mobile transaction will not be able to identify the validity of its accessed data items from the IR. For this case, all the cached data items will be considered invalid. Discarding all the cached data items for this case should not significantly affect the system performance since they are quite "old" and it is likely that most of them are already outdated. Furthermore, once the next IR has been received, the validity of all the cached data items can be identified. Therefore, the impact of disconnection on OUFO is small.

4.7. Broadcast index structure and doze mode operation

An important issue in the design of data broadcast algorithm is how to minimize the tune in time for getting broadcast data since this will affect the power consumption rate of mobile machines. As suggested from previous works on the design of data broadcast algorithm, an index for the organization of the broadcast cycle can be inserted at the head of each broadcast cycle. Each mobile client tunes into the index first. Based on the broadcast index, it can determine what data items are included in a cycle and when the required data items of its mobile transaction will be broadcast. Before the broadcast of the data items, it may turn to doze mode in order to minimize the power consumption of the mobile machine and wake up just before the broadcast of its required data items.

The re-broadcast mechanism of OUFO will affect the efficiency of broadcast index since the broadcast of a data item may be delayed due to re-broadcast. In order to minimize the impact of data re-broadcast on the broadcast schedule, we can set a maximum broadcast bandwidth for data re-broadcast, i.e., 5%, in each broadcast cycle. Therefore, the maximum additional delay of a data item from its original broadcast schedule will be bounded by this maximum broadcast bandwidth, i.e., is equal to 5% of a broadcast cycle.

However, setting a maximum re-broadcast bandwidth on the other hand will affect the correctness of OUFO since the conflict order cannot be reversed in case there is a conflict between an update transaction and a BT after the maximum re-broadcast bandwidth is reached. To ensure the correctness of OUFO, once the maximum re-broadcast bandwidth has been reached, the broadcast server may broadcast two additional pieces of information in the broadcast cycle. Firstly, the system broadcasts a signal periodically to inform the just awaken mobile clients that re-broadcast has stopped so that the currently executing mobile transactions need to check with the next IR for their accessed data items. The period for broadcasting the signal should be very short such that all mobile transactions will be informed before its commitment. Secondly, for any conflict between an update transaction and a BT, the broadcast server may broadcast the identities of the conflicting items instead of the values of the items as in the original OUFO. For those mobile clients, which are not in doze mode, they can determine the validation of the cached data items and the data items accessed by their transactions based on the broadcast data identity. Since the identity of is very small, the broadcast of the identities should not significantly affect the broadcast schedule. After ensuring the currency of its accessed data items, a mobile transaction may commit immediately without waiting for the IR.

Since the value of the maximum re-broadcast bandwidth will affect the power consumption in the mobile machines, it should be chosen based on the probability of data conflicts (number of re-broadcast), the amount of power saved in the doze mode as compared to the power consumption level in the active mode, and the total power supply of the mobile machines. If the power supply is not a highly critical issue, a larger maximum re-broadcast bandwidth may be used in order to gain the advantages of the re-broadcast mechanism of OUFO.

5. Performance studies

In this section, we report our simulation studies on the OUFO method as compared to two other efficient methods: the MV broadcast method (MV) and the periodic IR method. The reason for comparing OUFO with these two methods is that they represent two extremes between the tradeoff of concurrency and currency. As shown in [17], MV provides a high concurrency but a low currency while the IR provides a high currency but a low concurrency. Another reason for choosing MV is that it has a similar model assumption as OUFO, i.e., it assumes that the mobile transactions have deadlines. (Note that although shown in the performance results in Section 5.3, the performance of OUFO is better than MV in many aspects, we have no intention to claim that OUFO is better than MV. They are proposed for different types of systems. MV is for mobile computing systems where currency may not be a critical issue. Its main purpose is to tradeoff currency for consistency. What we would like to achieve in the simulation experiments is to investigate the performance characteristics of OUFO. Comparing with other methods, which are similar in assumptions, can make the analysis easier and useful.)

Following the definition in [17], the update transactions in MV will be written into the database at the end of a broadcast cycle. In MV, the number of versions to be broadcast is the number of versions created within the period of (current time−life span of a mobile transaction) to current time. Similar to the experiment setting in [17], 50% of the client cache is reserved for new versions of the data items and the remaining 50% are for old versions. The client cache uses the least recent policy for cache replacement. Note that although in [19] it was found that if old versions of data items were put in the secondary storage, the performance could be better, this way of storing the old versions may not be practical in many mobile machines, e.g., palm and handheld PC, since they do not have any secondary memory. The implementation of IR is similar to the invalidation scheme used in OUFO except that the generation of the IRs is fixed at the end of each broadcast cycle and all update transactions are performed at the end of a cycle.

We have implemented a simulation model using the CSIM-18, which is a simulation language based on the C programming language, to model the three methods, OUFO, MV and IR. Extensive simulation experiments have been performed to investigate the impact of different factors, such as update rate, data access pattern of the update transactions, cache size, database size, and network disconnection, on the performance of the three methods.

5.1. Simulation model

Since we assume that the major bottleneck of the system is the mobile network, in modeling the overhead of OUFO in the simulation model, we concentrate on its overhead on the mobile network, i.e., the overhead in broadcast of IRs and data re-broadcast. Basically, the model consists of three main entities and three processes as shown in Fig. 3. The three main entities are the broadcast server, the air media and the mobile clients. Two of the three processes are at the broadcast server: the broadcast process and the server update process. The last main process is the client process at each mobile client for simulating the mobile transactions.

Broadcast server: The database is located at the broadcast server. In order to simplify the set of model parameters, it is assumed that the data items have the same sizes. If they have different sizes, the broadcast time of the data items will be

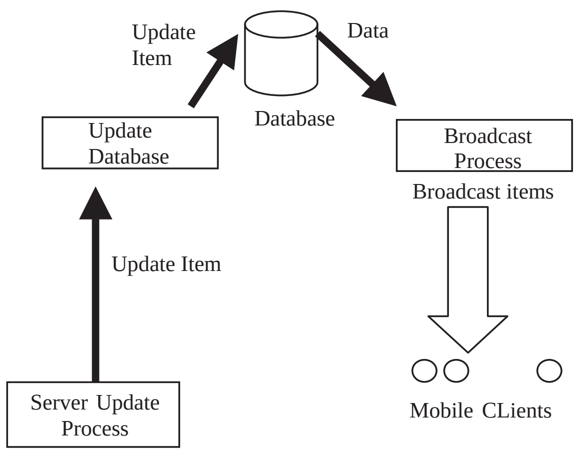

Fig. 3. Simulation model.

different. Each item has a unique identifier and an update time stamp indicating its last update time.

Air media: Air media is a collection of air channels. In our experiments, we assume that both scheduled broadcast items and re-broadcast items share the same broadcast channel.

Mobile clients: There are a large number of mobile clients in the system, which generates read-only transactions. Each client generates one mobile transaction at a time and will generate the next mobile transaction after an exponentially distributed think time upon the completion of a mobile transaction.

Broadcast process: The broadcast process defines the broadcast schedule, e.g., the selection of data items from the database for broadcast. We assume a flat broadcast disk model to simplify the model and analysis. Thus, a broadcast cycle includes all the data items in the database.

Server update process: The server update process generates update transactions. The data items to be updated by an update transaction are assumed to follow the Zipf distribution, which is commonly used in many previous studies in the area [1,17,19]. The server update process resolves data conflicts among the update transactions using 2PL.

Mobile client process: The mobile client process generates firm deadline mobile transactions. A transaction will be aborted once its deadline has been missed. If the transactions have soft deadlines, they will exist in the system even if they have missed their deadlines. It is expected that the performance of OUFO will be similar if we use soft deadline transactions. Although the restart rate may be higher as a transaction may exist longer in the system, the restart overhead will be similar to the case with firm deadline transactions since the clients maintain recently accessed data at their caches. Restarted transactions can easily find their required data at the cache and do not need to get them from broadcast cycles again.

Each transaction consists of a sequence of operations and each operation requires access to one data item. It is assumed that the data access pattern of the operations follows the Zipf distribution. Once generated, the mobile transaction will try to access the required data item of its first operation from the cache. If it does not find the data item at the cache, it will listen to the broadcast cycle continuously for the data item. After getting its required data item, the transaction will access the CPU for computation. Then, it will process its next operation. When a mobile transaction has completed all its operations, it will commit.

5.2. Model parameters and performance measures

5.2.1. Model parameters

Table 1 lists the model parameters and their baseline values. Some of the parameter values are chosen based on the values used in previous studies in the area [17,19]. The purpose is to facilitate comparison with results from related works. Note that the main purpose of the simulation study is not to investigate the performance of the proposed method for a particular mobile computing application. Instead, it is to investigate the performance characteristics of the proposed method as compared with other related methods. In the simulation experiments, we vary the parameter over a wide range of values to get a more complete picture on the performance of the proposed method.

5.2.2. Performance measures

The primary performance measures are *miss rate*, *mean response time* and *stale access rate*. Miss rate is defined as the number of mobile transactions, which miss their deadlines (aborted) divided by the total number of mobile transactions generated. It indicates the capability of meeting the timing requirements of the mobile transactions. Note that unlike conventional real-time systems, in addition to meeting the deadline constraints, it is also important to minimize the mean response time. Stale access rate is defined as the number of stale data accesses, e.g., reading a data value, which is different from that of the latest updated value, divided by the total number of data accesses. It is an important measure for the currency of the data items observed by the mobile transactions. Of course, it is preferred to minimize the stale access rate.

In addition to the primary measures, we also measure other statistics, such as the *cache hit rate*, *broadcast overhead*, *restart rate* and *broadcast hit rate*, to investigate the performance characteristics of the methods. Broadcast overhead captures the amount of broadcast bandwidth consumed by the methods. Broadcast overhead in OUFO is the percentage of bandwidth used for re-broadcasting data items due to data conflicts and the overhead

Table 1
Model parameters ad their baseline values

Parameter	Baseline value
Database size	1000 data items
No. of mobile clients	100
Broadcast rate	20 data items/s
Cache size	50 data items
Cache replacement scheme	LRU
Data access distribution (both mobile and update transactions)	Zipf distribution
Degree of skew in access distribution	1.0
Offset between the data access distribution of mobile and update transactions	10%
No. of operations in a mobile transaction	1–4
No. of write operations in an update transaction	1–2
Invalidation report generation period (for OUFO and IR only)	50 s
Report duration	1000 s
Life span of a mobile transaction	200 s
Mean think time of a mobile client	10 s
Mean inter-arrival time of update transactions	0.1–4 s
Disconnection probability	0.01 and 0.1
Disconnection time	100 ms

for broadcasting the IR. In MV, the broadcast overhead is the percentage of broadcast bandwidth used for broadcasting all versions of data items, except the newest version. Restart rate is defined as the number of restarts of mobile transactions over the total number of mobile transactions completed. In OUFO and IR, a mobile transaction may be restarted after checking with an IR or it has accessed an invalid data item. In OUFO, a mobile transaction may also be restarted if any of its accessed data item is re-broadcast. Broadcast hit rate is calculated as the number of data items obtained directly from the broadcast cycles by the mobile transactions divided by the simulation length. If broadcast hit rate is high, the average waiting time for data items from the broadcast cycles will be smaller.

5.3. *Performance results and discussion*

5.3.1. *Impacts of update workload*

Figs. 4–10 show the impact of update interval on the performance of the three methods, OUFO, MV and IR, when different skew coefficients (0.5

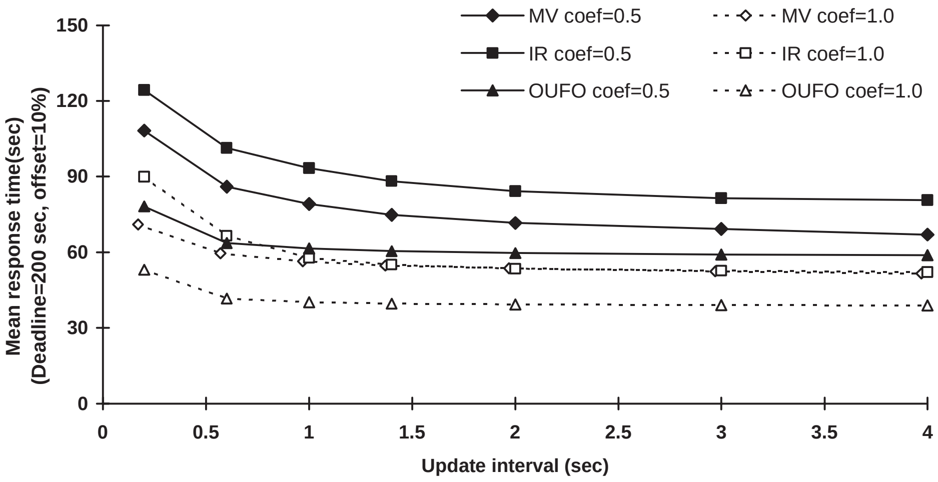

Fig. 4. Impact of update workload on mean response time at different skew coefficients.

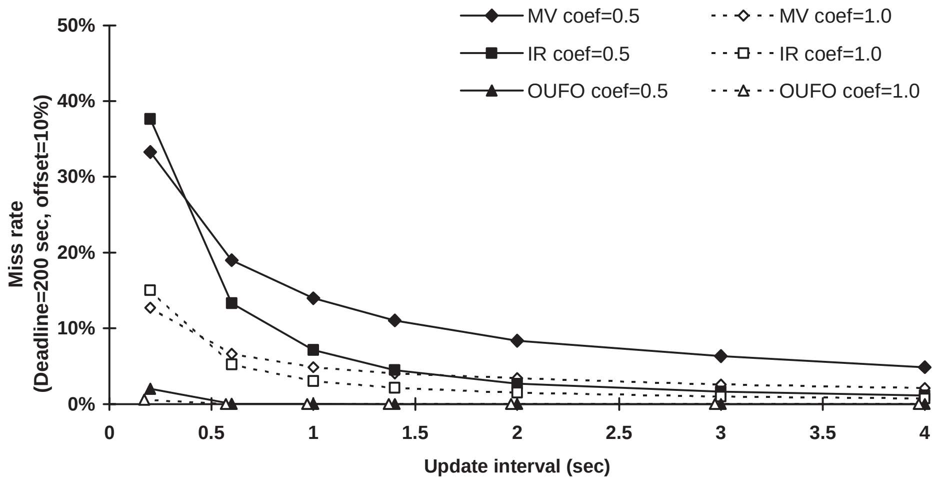

Fig. 5. Impact of update workload on miss rate at different skew coefficients.

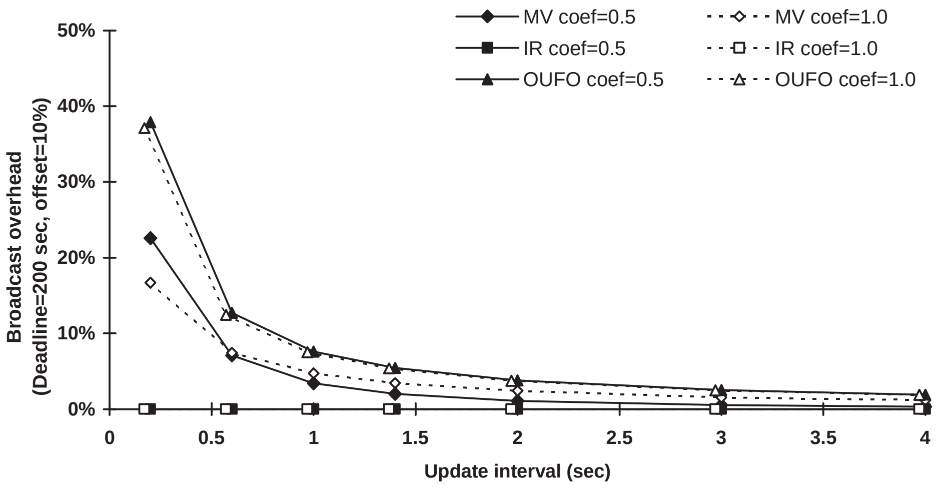

Fig. 6. Impact of update workload on broadcast overhead at different skew coefficients.

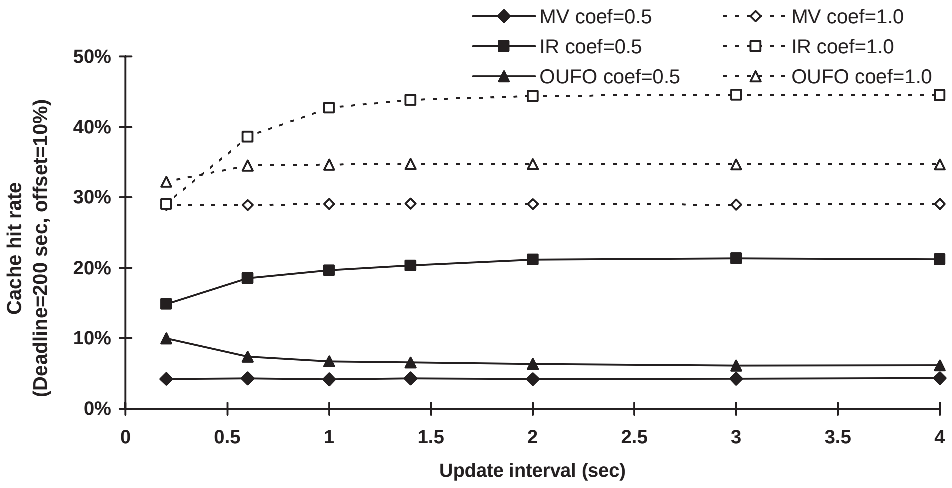

Fig. 7. Impact of update workload on cache hit rate at different skew coefficients.

and 1.0) are used for the Zipf distribution which is adopted by both update transactions and mobile transactions. An offset of 10% is set between the Zipf distributions for the mobile transactions and the update transactions so that they do not have exactly the same set of hot data items. Note that increasing the skew coefficient decreases the number of hot data items. As shown in Figs. 4 and 5, the mean response time and miss rate of the three methods decrease gradually with an increase in update interval, respectively. This is consistent with our expectation. Increasing the update interval reduces the update workload and the data conflict probability between the update and mobile transactions. The consequence will be lighter re-broadcast workload in OUFO, lower MV broadcast overhead in MV, and smaller number of restarts in IR. These can be observed in Figs. 6 and 9.

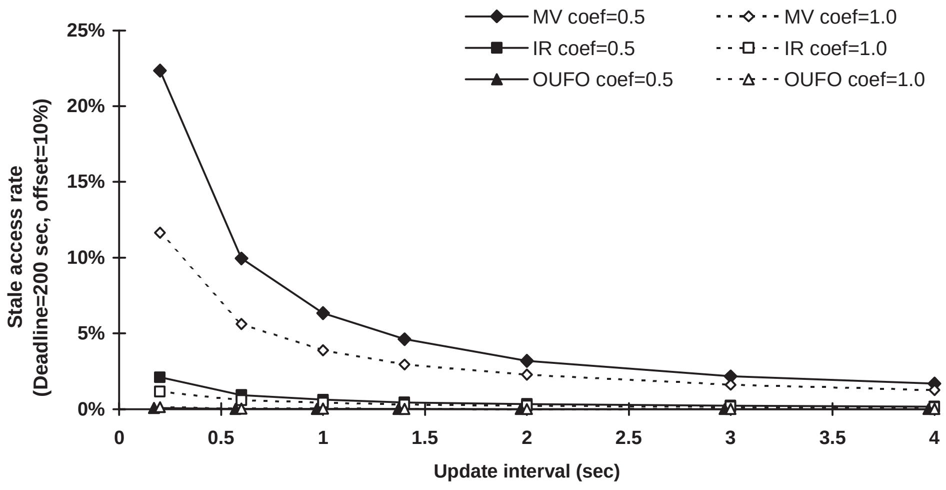

Fig. 8. Impact of update workload on stale access rate at different skew coefficients.

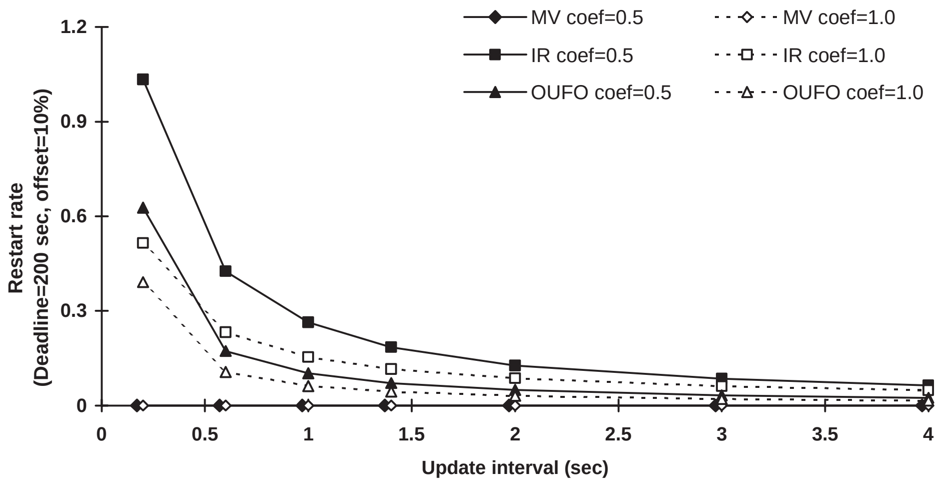

Fig. 9. Impact of update workload on restart rate at different skew coefficients.

As depicted in Figs. 4 and 5, the performance (mean response time and miss rate) of OUFO is consistently much better than that of MV and IR although its broadcast overhead is higher as shown in Fig. 6 especially when the update workload is heavy. The better performance of OUFO as compared with MV is mainly due to (1) a higher cache hit rate (as shown in Fig. 7) and (2) a much higher broadcast hit rate as shown in Fig. 10. The better performance of OUFO as compared with IR is due to lower restart rate as shown in Fig. 9 and higher broadcast hit rate as shown in Fig. 10. In MV, multiple versions of a data item are maintained at the client cache. This can seriously reduce the number of different data items at the client caches, leading to a much smaller cache hit rate. The broadcast of multiple versions of data items also seriously increases the broadcast overhead. The much better broadcast hit rate in OUFO as compared to MV and IR is due to the

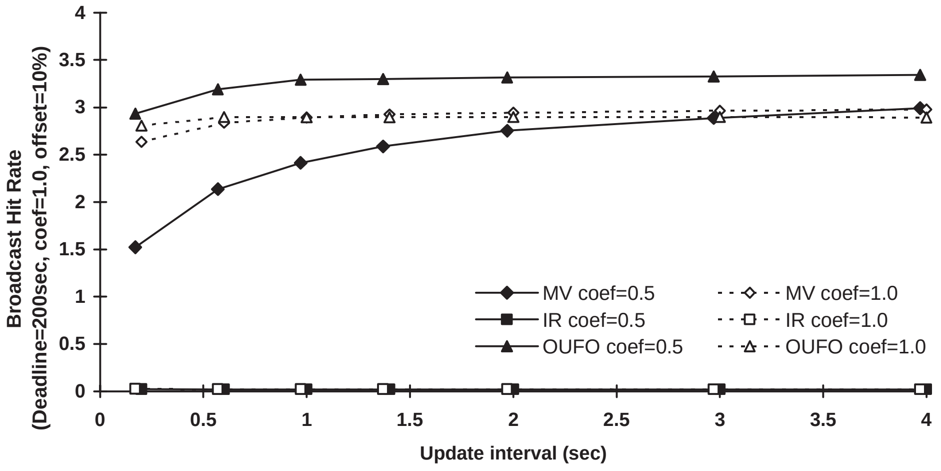

Fig. 10. Impact of update workload on broadcast hit rate at different skew coefficients.

rebroadcast mechanism in OUFO. Although rebroadcasting a data item introduces additional broadcast overhead, at the same time, it increases the broadcast hit rate especially when the data access patterns of update and mobile transactions are similar, i.e. both are Zipf distributions. A rebroadcast item may be waiting by other mobile transactions. Thus, the average waiting time for data items will become smaller. As shown in Figs. 7 and 10, the cache hit rate of IR is slightly higher than that of OUFO but its broadcast hit rate is very low (close to zero) indicating that the mobile transactions obtain most of their required data from the cache instead from data broadcast. Due to the long waiting time for data items, the restart rate of IR is much higher than that of OUFO. Note that the cost of transaction restart in IR is much higher. A restarted transaction has to wait for its data items from the broadcast cycle again since the cached copy is invalidated.

In addition to smaller miss rate and mean response time, another important advantage of OUFO over MV is that it can provide a higher data currency to the mobile transactions as shown in Fig. 8. The stale access rate of OUFO remains zero while the stale access rate of MV is up to 25% when the workload is heavy and the skew coefficient is equal to 0.5. This indicates that many mobile transactions observe "old" values under MV.

Although the performance of IR is similar to MV when the skew coefficient equals 1.0, IR is much better than MV when the skew coefficient is equal to 0.5. This is mainly due to a much higher cache hit rate (Fig. 7) and a lower broadcast overhead (Fig. 6), although its restart rate is higher than MV (Fig. 9). The stale access rate of IR is also much lower as compared with MV as shown in Fig. 8 since the cache only maintains the most update versions of the data items. Outdated versions are invalidated by IR periodically.

Figs. 4 and 5 also show that the performance of OUFO, IR and MV is better when the skew coefficient is larger. The skew coefficient affects the number of hot data items. Reducing the skew coefficient (e.g., from 1.0 to 0.5) increases the number of hot items. This reduces the cache hit rate as shown in Fig. 7 and increases the average data access delay of mobile transactions.

Figs. 11 and 12 show the performance of the three methods when the offset of the Zipf functions for the update and mobile transactions is set to zero, e.g., the two types of transactions have the same set of hot data items. Consistent with the previous results, the performance (mean response time and miss rate) of OUFO is consistently better than that of MV and IR as shown in Figs. 11 and 12. However, the performance of IR becomes worse than MV due to large number of transaction restarts. Although the

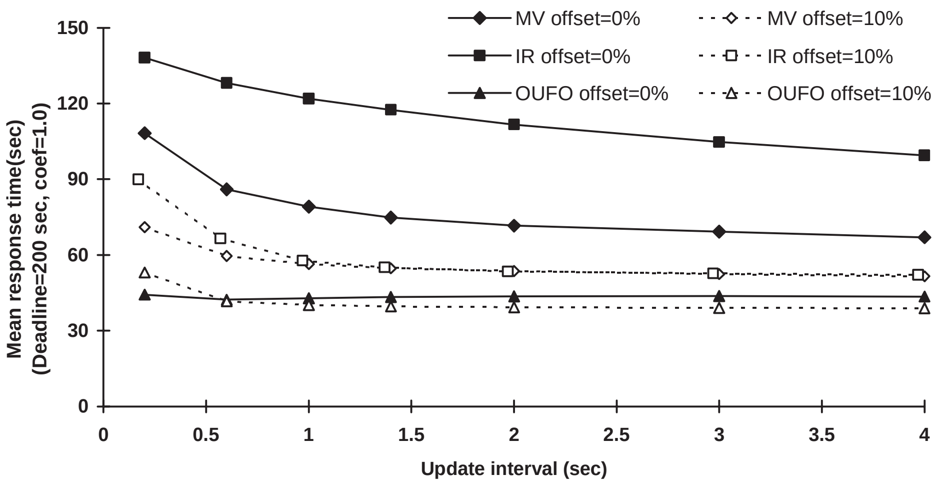

Fig. 11. Mean response time at different offsets between the Zipf distributions.

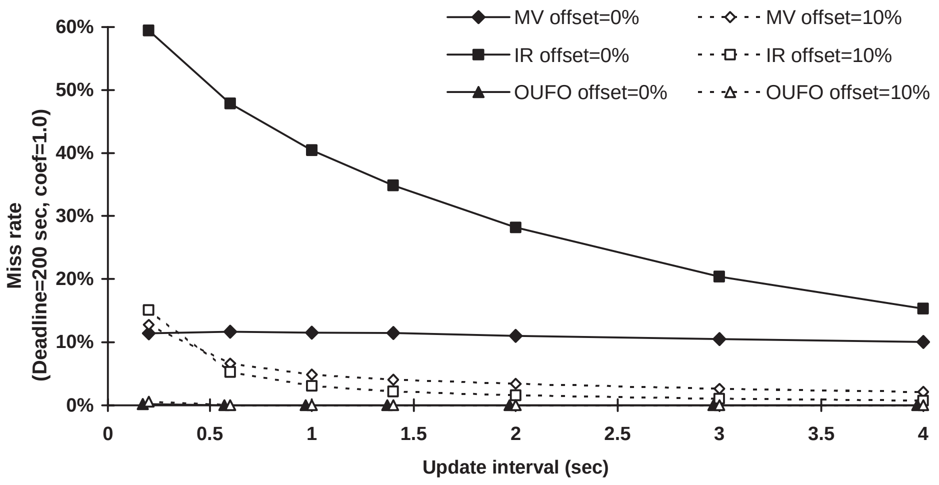

Fig. 12. Miss rate at different offsets between the Zipf distributions.

restart rate of OUFO is also high when the offset is set to be zero, the cost of transaction restarts in OUFO is very low since a restarted transaction can always find its previously accessed data items at the client cache. Furthermore, increasing the degree of overlap between the access patterns of mobile transactions and update transactions at the same time increases the broadcast hit rate of OUFO.

Figs. 13 and 14 show the results when the length of a mobile transaction is increased from 1–4 operations to 4–8 operations. As shown in the figures, increasing the transaction length increases the miss rate and mean response time since a mobile transaction has to wait longer to get all its required data items. Consistent with the results in previous figures, OUFO gives the best performance. Increasing the length of the transactions, the overhead of the methods (MV and OUFO) and the restart probability of a transaction (OUFO and IR) will be higher. Although the number of transaction restarts is

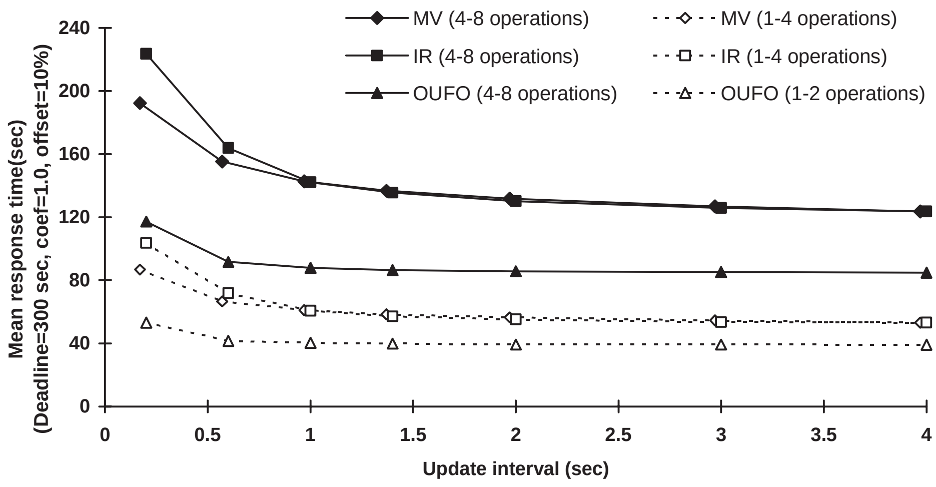

Fig. 13. Mean response time at different lengths of mobile transactions.

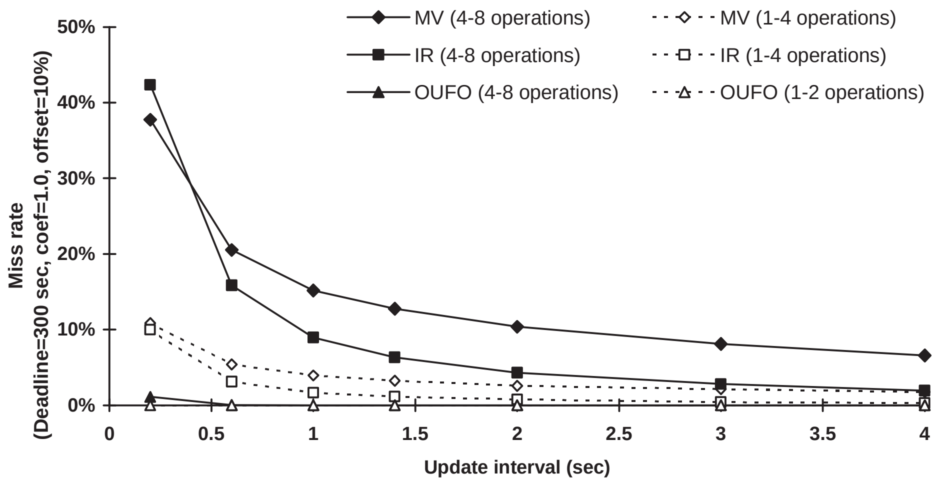

Fig. 14. Miss rate at different lengths of mobile transactions.

also higher under OUFO when the transaction length is longer, this does not seriously affect the performance since the restart cost is low as a restarted transaction can find its required data items from the cache.

5.3.2. Impact of cache size and database size

Figs. 15–18 show the performance of the three methods at different client cache sizes. As expected, increasing the cache size improves the performance of the methods as shown in Figs. 15 and 16. It is mainly due to a higher cache hit rate as shown in Fig. 17. Consistent with the results in the last sub-section, the performance of OUFO is significantly better than that of MV and IR as shown in Figs. 15 and 16, e.g., smaller mean response time and lower miss rate. The better performance of OUFO is again due to higher cache hit rate (Fig. 17), lower broadcast overhead (Fig. 18) and higher broadcast hit rate. The

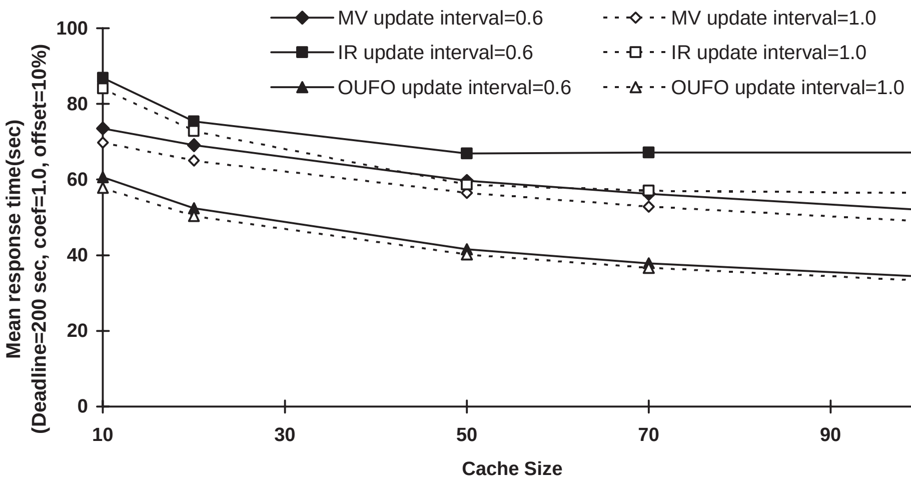

Fig. 15. Mean response time at different cache sizes.

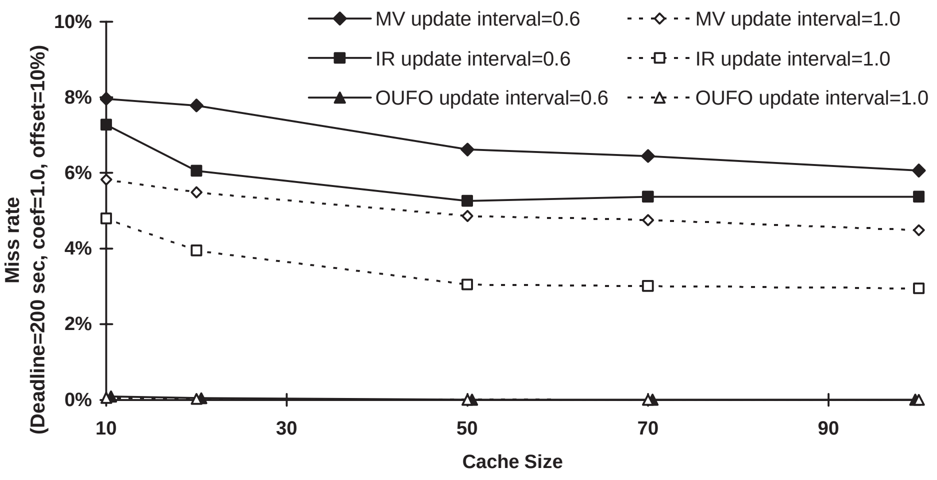

Fig. 16. Miss rate at different cache sizes.

differences in their performance are greater when the update workload is heavier. The poor mean response time of IR is mainly due to large number of restarts and long waiting time for data items from data broadcast.

Increasing the database size increases the length of a broadcast cycle as we are using a flat broadcast disk (where a broadcast cycle includes all the data items in the database.) At the same time, the number of hot data items will also be larger due to a larger database. Thus, the conflict probability will be lower. Figs. 19–21 show the results when the database size is increased from 1000 to 2000 data items. It can be observed that the performance of the methods degrades when a larger database is used. It is because: (1) the mobile transactions have to wait longer for their data items if they cannot find their required data items in the cache due to a longer broadcast cycle; and (2) a lower cache hit rate as the number of hot data items is larger (Fig. 21). The performance of OUFO remains consistently

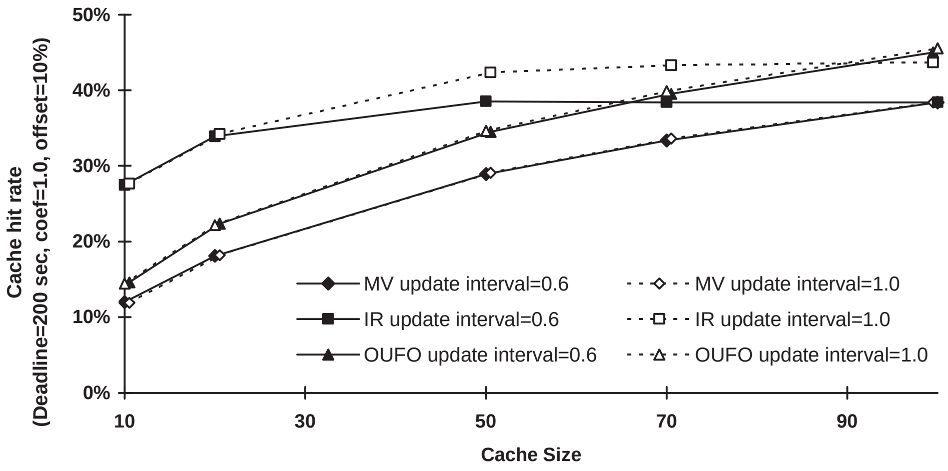

Fig. 17. Cache hit rate at different cache sizes.

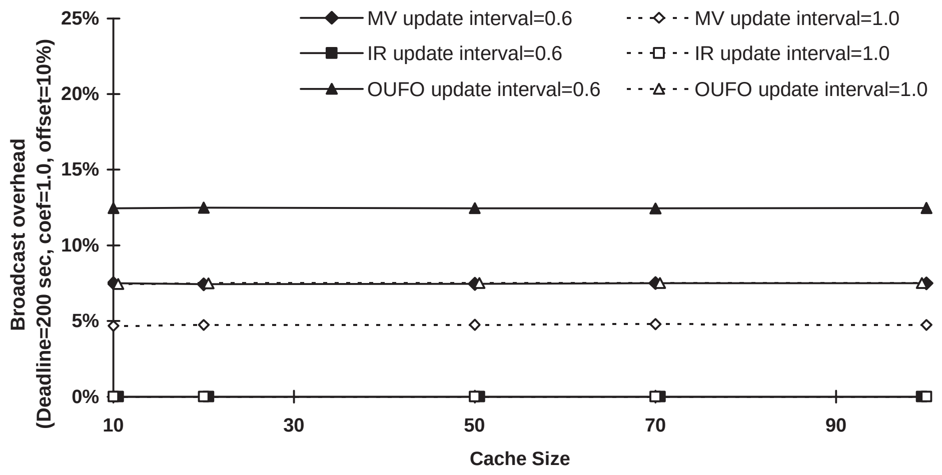

Fig. 18. Broadcast overhead at different cache sizes.

better than MV and IR in mean response time, miss rate and stale access rate even if the database size is larger.

5.3.3. Impact of network disconnection

In this section, we investigate how network disconnection affects the performance of the three methods. In OUFO, once a disconnection has occurred, the mobile client will consider all its cached data items to belong to the unknown version group. All the executing mobile transactions are required to validate with the IR before they can commit. The impact of network disconnection does not affect the correctness of MV. However, under MV, a mobile transaction cannot receive new data items from the broadcast server. This will affect the probability of finding the right version of a data item. For IR, if network disconnection occurs while an IR is receiving, a mobile transaction will be required to validate its accessed data items with the next validation report after the reconnection.

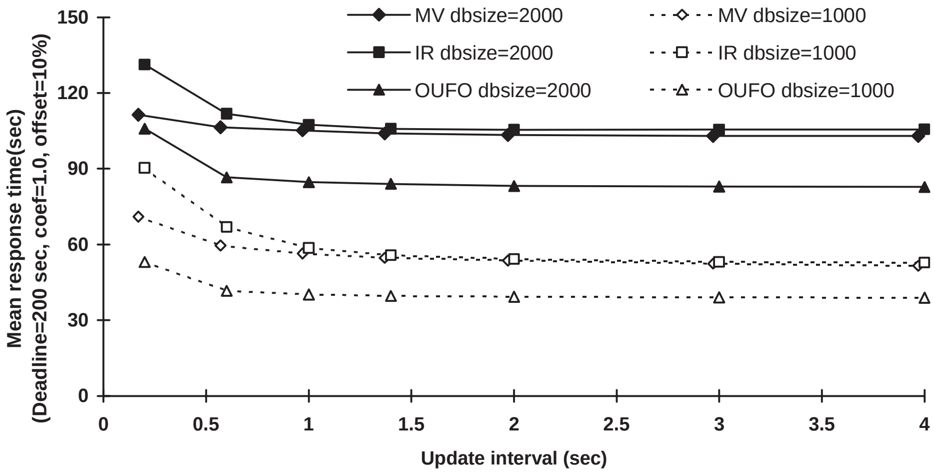

Fig. 19. Mean response time at different database sizes.

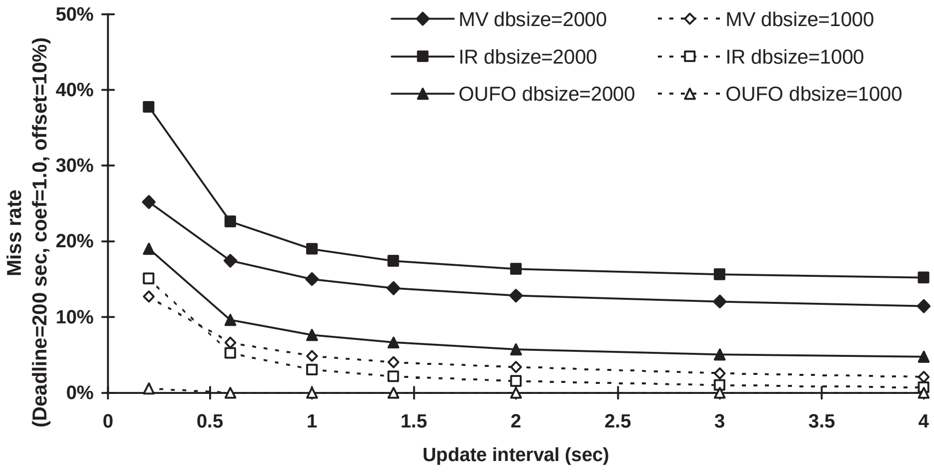

Fig. 20. Miss rate at different database sizes.

To model network disconnection, we introduce two new parameters, the disconnection probability and disconnection time. In the experiments, we concentrate on the involuntary disconnection since it is assumed that voluntary disconnection will only occur after the completion of mobile transactions. The disconnection time is set to be 100 ms. In our model, a mobile client may disconnect from the network following the disconnection probability after receiving a broadcast item. Two sets of experiments have been performed: high disconnection probability and medium disconnection probability. Figs. 22 and 23 show the mean response time and miss rate, respectively, when a low disconnection probability, 0.01, and a higher disconnection probability, 0.1, are used. As shown in Figs. 22 and 23, the performance of the three methods is worse than the case without disconnection and the performance of the three methods is poorer when the disconnection probability is higher. However, the miss rate and mean response time of OUFO are still significantly smaller than that of MV and

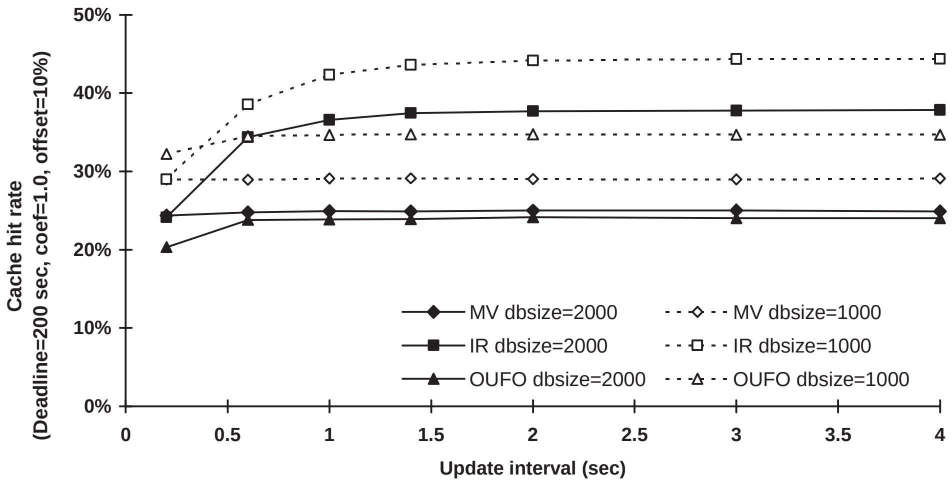

Fig. 21. Cache hit rate at different database sizes.

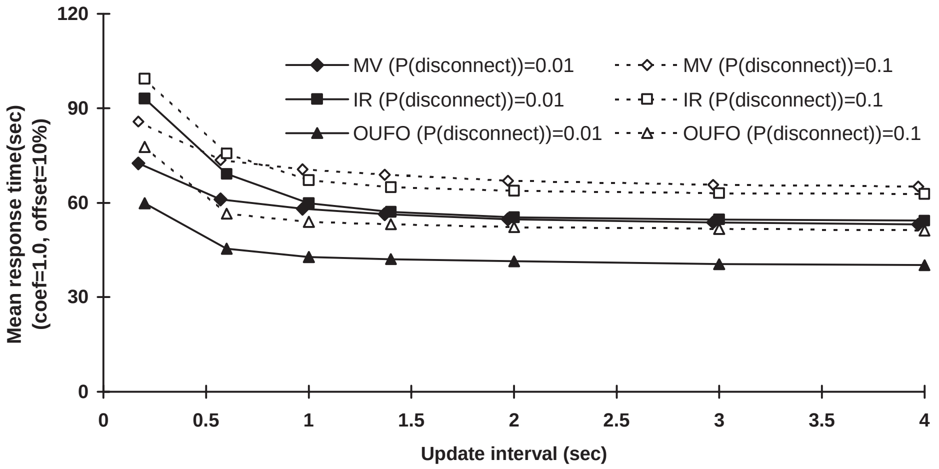

Fig. 22. Mean response time at different disconnection probabilities.

IR even when the disconnection probability is high, i.e., 0.1.

6. Conclusions

Data broadcast has been shown to be an efficient method for disseminating data items to transactions generated by mobile clients. Although the research in the design of broadcast algorithms has received a lot of attention in previous years, the concurrency control issue has been largely ignored. If broadcast of data items and execution of update transactions (which are important for maintaining the validity of the data items at the database) are uncontrolled, mobile transactions may observe inconsistent data values.

In this paper, we study the issue of concurrency control between update transactions and mobile transactions, which consist of ordered read-only operations. Our proposed method, called ordered update first with order (OUFO), is not only

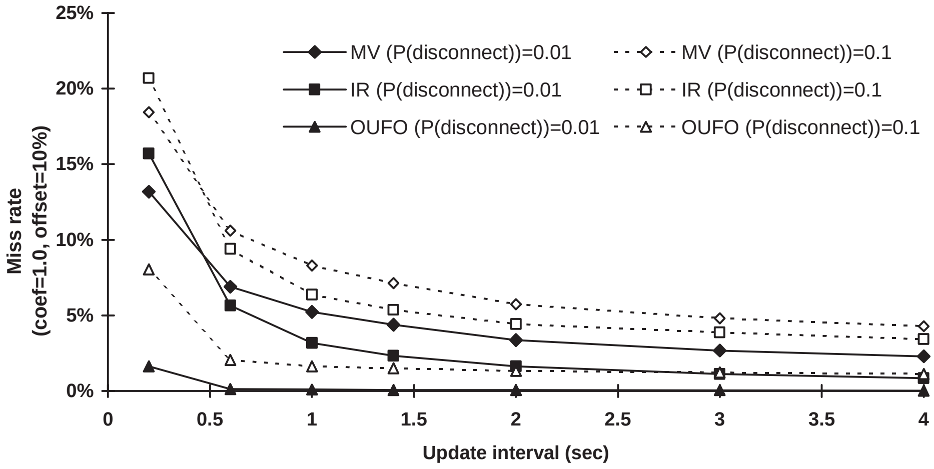

Fig. 23. Miss rate at different disconnection probabilities.

simple, but can also maintain the schedule between update and mobile transactions to be serializable and at the same time maximize the currency of the data items observed by the mobile transactions. OUFO can easily be applied to different broadcast algorithm and its impact on the mobile clients, especially in cache data management, is minimal. This means that OUFO can be implemented easily in most mobile broadcast systems.

In OUFO, an efficient cache management scheme is adopted to minimize the cost of transaction restarts due to data conflicts with update transactions and reduce the data access delays. It combines both data re-broadcast and explicit IRs to ensure that all the accessed cached data items are valid. Although at first sight it might appear that data re-broadcast will increase the broadcast overhead significantly, we have found that in practice the data items being re-broadcast are often hot data items and the re-broadcast actually has the desirable side effect of reducing the waiting time of other mobile transactions not involved in the data conflict. Thus, OUFO can still achieve good performance even when data conflict probability is high.

In order to investigate the performance characteristics of the OUFO method, a simulation model of a mobile computing system with data broadcast has been implemented. Experiments are performed to investigate the impact of different system parameters on its performance as compared with the MV broadcast method as well as the periodic invalidation method. The results show that OUFO significantly outperforms the other two efficient methods in terms of mean response time of mobile transactions and miss rate. Another important advantage is that the data items under OUFO are generally much more current than those under MV broadcast method. Although disconnection may affect the efficiency of OUFO, performance results indicate that OUFO still give a much better performance than MV and IR when the disconnection probability is high.

Acknowledgements

We would like to thank the reviewers of the COMAD'2000 for their valuable comments and suggestions on the earlier version of the paper which appeared in COMAD'2000.

References

[1] S. Acharya, M. Franklin, S. Zdonik, Balancing push and pull for data broadcast, in: Proceedings of the ACM SIGMOD, Tucson, Arizona, May 1997.
[2] M. Franklin, S. Zdonik, Data in your face: push technology in prospective, in: Proceedings of the 1998 ACM SIGMOD Conference, Seattle, 1998.

[3] T. Imielinski, B.R. Badrinath, Mobile wireless computing: challenges in data management, Commun. ACM 37 (10) (1994).

[4] E. Pitoura, B. Bhargava, Maintaining consistency of data in mobile distributed environment, in: Proceedings of the 15th International Conference on Distributed Computing Systems, 1995, pp. 404–413.

[5] H.W. Leung, Concurrency control for data broadcasts in mobile computing systems, M.Phil. Thesis, Department of Computer Science, City University of Hong Kong, 2001.

[6] S. Hameed, N.H. Vaidya, Efficient algorithms for scheduling single and multiple channel data broadcast, Technical Report 97-002, Department of Computer Science, A&M University, Texas, February 1997.

[7] H.V. Leong, A. Si, Data broadcasting strategies over multiple unreliable wireless channels, in: Proceedings of the Fourth International Conference on Information and Knowledge Management, November 1995, pp. 96–104.

[8] H.V. Leong, A. Si, Database caching over the air-storage, Comput. J. 40 (7) (1997) 401–415.

[9] P. Xuan, O. Gonzalez, J. Fernandez, K. Ramamritham, Broadcast on demand: efficient and timely dissemination of data in mobile environments, in: Proceedings of the Third IEEE Real-Time Technology Application Symposium, 1997.

[10] R. Srinivasan, C. Liang, K. Ramamritham, Maintaining temporal coherency of virtual data warehouses, in: Proceedings of Real-time Systems Symposium, Spain, 1998, pp. 60–70.

[11] O. Ulusoy, Real-time data management for mobile computing, in: Proceedings of the International Workshop on Issues and Applications of Database Technology (IADT'98), Berlin, Germany, July 1998.

[12] O. Wolfson, P. Sistla, S. Chamberlain, Y. Tesha, Updating and querying databases that track mobile units, J. Distributed Parallel Databases 7(3) (1999).

[13] E. Kayan, O. Ulusoy, An evaluation of real-time transaction management issues in mobile database systems, Comput. J. 42(6) (1999), 501–510.

[14] P.A. Bernstein, V. Hadzilacos, N. Goodman, Concurrency Control and Recovery in Database System, Addison-Wesley, Reading, MA, 1987.

[15] J. Shanmugasundaram, A. Nithrakashyap, R. Sivasankaran, K. Ramamritham, Efficient concurrency control for broadcast environments, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, Philadelphia, June 1–3, 1999.

[16] Kam-Yiu Lam, Mei-Wai Au, Edward Chan, Broadcasting consistent data to read-only transactions from mobile clients, Comput. J. (2001), in preparation.

[17] E. Pitoura, P.K. Chrysanthis, Scalable processing of read-only transactions in broadcast push, in: Proceedings of the International Conference on Distributed Systems, May 1999.

[18] E. Pitoura, Supporting read-only transactions in wireless broadcasting, in: Proceedings of the DEXA'98 Workshop on Mobility in Databases and Distributed Systems, August 1998.

[19] E. Pitoura, P.K. Chrysanthis, Exploiting versions for handling updates in broadcast disks, in: Proceedings of the Very Large Data Base Conference, Edinburgh, UK, September 1999.

[20] S. Acharya, R. Alonso, M. Franklin, S. Zdonik, Broadcast disks: data management for asymmetric communications environments, in: Proceedings of the ACM SIGMOD, San Jose, CA, 1995.

[21] J. Fernandez, K. Ramamritham, Adaptive dissemination of data in real-time asymmetric communication environments, in: Proceedings of the Euromicro Conference on Real-Time Systems, June 1998.

[22] J. Fernandez, K. Ramamritham, Adaptive dissemination of data in asymmetric communication environment, ACM Mobile Networks Appl. J., in preparation.

[23] A. Datta, A. Celik, J. Kim, D.E. VanderMeer, Adaptive broadcast protocol to support power conservant retrieval by mobile users, in: Proceedings of the International Conference on Data Engineering, 1997, pp. 124–133.