

Mobile Real-Time Read-Only Transaction Processing in Broadcast Disks

HONG-YA WANG¹, GUO-QIN NING², GUO-HUI LI³ AND KAM-YIU LAM⁴

¹*School of Computer Science and Technology*

Donghua University

Shanghai 200016, P.R.C.

²*Department of Information Technology*

Central China Normal University

Hubei 430079, P.R.C.

³*College of Computer Science and Technology*

Huazhong University of Science and Technology

Hubei 430074, P.R.C.

⁴*Department of Computer Science*

City University of Hong Kong

Kowloon, Hong Kong

Data management issues in mobile computing environments have got lots of concerns of relevant researchers. Among these research topics, data broadcast has been extensively investigated due to its advantages such as scalability and bandwidth effectiveness. While plenty of works have been done on this subject, it is still less touched when data broadcast is used to deliver frequently updated real-time data to read-only transactions with deadlines, which we call *updates dissemination* in this paper. Existing updates dissemination protocols are unsuitable for mobile real-time read-only transaction processing since they neglect the time constraints on both data and transactions. In this paper, we first identify the data dissemination requirements for read-only transaction processing in mobile real-time computing environments by formally defining several performance objectives. Then a novel updates dissemination protocol called *hybrid forward multi-version data broadcast* is proposed. In *hybrid forward multi-version data broadcast*, a set of new techniques including reducing the length of consistency intervals, instantly broadcasting updates policy and broadcast on-demand are incorporated seamlessly. Simulation results show that the proposed protocol can provide higher data currency and lower miss rate compared with existing updates dissemination protocols and is more appropriate for mobile real-time read-only transaction processing in broadcast disks.

Keywords: updates dissemination, broadcast on-schedule, broadcast on-demand, mobile real-time read-only transaction, transaction processing

1. INTRODUCTION

Thanks to rapid advances in wireless communication technology, many novel mobile database applications have emerged to meet the need of accessing data resources anywhere and anytime. Among these applications, mobile read-only transaction processing that aims at providing consistent data to read-only transactions issued from mobile

Received October 11, 2004; revised December 21, 2004; accepted March 23, 2005.
Communicated by Yu-Chee Tseng.

clients has got lots of concerns of relevant researchers [1-7]. Owing to the inherent limitation of wireless communication network, the bandwidth of the downstream channel from the database server to mobile clients is much larger than that of the upstream channel. Therefore, instead of pull-based data dissemination methods, data broadcast has been widely accepted as an efficient way for mobile clients to get their needed data. In [1, 2, 8, 10] efficient access methods and broadcast organization structures are investigated, which focus on minimizing the access delay as well as the tune-in time. In [14, 16, 17] different data selection and broadcast scheduling algorithms, which have the same performance objectives as above, are proposed and evaluated. Some researchers have proposed hybrid approaches in which periodic broadcast of data is combined with support for explicit client requests and simulation experiments show that the performance of the hybrid approach depends on careful allocation of bandwidth for the two types of data dissemination [3, 6, 13]. All above research results indicate that data broadcast can better utilize the limited bandwidth of mobile computing systems and it is particularly suitable for dissemination of large volume of data to a large number of mobile clients. When data broadcast is used to propagate frequently updated data to read-only transactions issued from mobile clients, we call it *updates dissemination*.

To the best of our knowledge, updates dissemination is first addressed in [1] and the authors discuss the tradeoffs between data currency and performance issues. However, transaction semantics is not supported and the consistency criterion is relatively relaxed. Transactional support is introduced in the Datacycle architecture and the serializability is taken as the correctness criterion, while special hardware is needed to detect changes of values read by transactions [12]. In [7] a new correct criterion is proposed and a control matrix based method is used to ensure the consistency of mobile read-only transactions as well as mobile update transactions. Recently two efficient updates dissemination protocols, the multiversion data broadcast (MVB) and the invalidation method (IM), are proposed in [4, 5]. In the invalidation method an invalidation report, which consists of a list including all data items that are updated at the broadcast server during the previous broadcast cycle, is periodically broadcast before each broadcast cycle. The validity of a mobile transaction is guaranteed by checking the invalidation report. A transaction has to be restarted in case any of its accessed data items is invalidated. In multiversion data broadcast the server broadcasts previous versions of data items in addition to the committed version of the data items at the last broadcast cycle. When 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 data items with the largest version number that is smaller than or equal to the data version of the first operation. By allowing a transaction to read older versions of data, data consistency can be ensured at the expense of data currency.

Though plenty of works have been done on data or updates dissemination in broadcast disks, the majority of existing updates dissemination protocols do not take the time constraints into consideration whereas data dissemination methods referring to real-time characteristics do not support the transactional semantics [6, 13]. *Mobile real-time read-only transaction (MRRT)* processing, which not only requires consistent data access to database servers but also has real-time constraints on both data and transactions, has not received adequate attention even though its significance for applications such as stock and traffic information dissemination [6, 9, 11]. The most striking characteristics of such

applications are that the values of data are highly dynamic and transactions are associated with deadlines.

In this paper, we investigate the problems of disseminating consistent data to mobile read-only transactions with deadlines. After deeply analyzing the characteristics of *MRRT* processing, two performance metrics, the system staleness and the miss rate, are formally defined. Subsequently, a new updates dissemination protocol called *hybrid forward multi-version data broadcast* is proposed. *HFMVB* not only guarantees the consistency of read-only transactions, it also provides higher data currency and lower miss rate by shortening the consistency interval, instantly broadcasting updates and utilizing broadcast on-demand. Simulation results show that *HFMVB* has better performance compared with existing updates dissemination protocols and is more appropriate for *MRRT* processing.

The rest of the paper is organized as follows. Section 2 presents the motivation and preliminaries for *MRRT* processing. Section 3 introduces the details of *HFMVB*. Section 4 presents the performance evaluation of the proposed updates dissemination protocols. Finally, the conclusion is made in section 5.

2. THE PROBLEM

2.1 Motivation

Two pioneering updates dissemination protocols, the *multiversion data broadcast* and the *invalidation method*, are proposed to provide consistent data items to read-only transactions in data broadcast environments in [4, 5]. For *MVB* and *IM*, the main performance concerns are the average response time and the restart rate. However, for an important category of applications that not only require consistent access to data items but also pose time constraints on both data and completion times, different performance objectives are required from traditional read-only transaction processing. For such applications, the usefulness of execution results may be affected negatively if all data requirements cannot be satisfied before assigned deadlines. At the same time, whether the data items read by transactions are stale (have been updated at the server) has an impact on the usefulness of transactions as well. This kind of applications needs the supports of updates dissemination protocols for mobile real-time read-only transaction processing. Existing updates dissemination protocols such as *MVB* and *IM* do not take time requirements into consideration. In view of shortcomings of preceding updates dissemination protocols, it is necessary to investigate updates dissemination protocols that take new requirements for *MRRT* processing into account.

2.2 Notations and Definitions

We assume that the database is a finite set composed of a number of data items, that is, $DB = \{d_i \mid i = 1, 2, \dots, n\}$.

Definition 1 The value of data item d at time instance τ is called the state of d at τ , denoted by $S_\tau(d)$. The states of all data items in DB at time instance τ is called the state of DB at τ , denoted by $S_\tau(DB) = \{S_\tau(d_i) \mid i = 1, 2, \dots, n\}$.

Definition 2 Let $S_{\tau_1}(d)$ and $S_{\tau_2}(d)$ be the states of d at τ_1 and τ_2 respectively. If condition $(\tau_1 < \tau_2) \wedge (S_{\tau_1}(d) \neq S_{\tau_2}(d))$ holds, we say that $S_{\tau_2}(d)$ is more current than $S_{\tau_1}(d)$, denoted by $S_{\tau_2}(d) >_c S_{\tau_1}(d)$. Let T be an mobile real-time read-only transaction that committed at time instance τ_c and $RS(T)$ is the read set of T . $S^T(d)$ denotes the state of d when T reads d . We say that T is consistent if and only if:

$$\exists \tau (\tau \leq \tau_c) \wedge (\forall d (d \in RS(T)) \wedge (S^T(d) \in S_{\tau}(DB))).$$

T is consistent only if data items in its read set form a subset of a consistent state of DB at time instance τ early than τ_c [4]. Guaranteeing the consistency of transactions is essential for many applications such as stock and traffic information dissemination. So we strictly follow the consistency criterion and do not consider relaxing it to gain the improvements in other performance metrics in this paper.

Definition 3 If T is consistent and condition $(d \in RS(T)) \wedge (S_{\tau_c}(d) >_c S^T(d))$ holds, we say that d is stale for T . Let $SR(T)$ be the set of stale data items read by T , that is, $SR(T) = \{d \mid d \in RS(T) \wedge S_{\tau_c}(d) >_c S^T(d)\}$. The staleness of T , denoted by ST_T , is defined as $|SR(T)| / |RS(T)|$. The system staleness, denoted by SS , is defined as $\sum ST_{T_i} / |\{T_i\}|$, where $\{T_i\}$ is the finite set of mobile real-time read-only transactions having committed without violating the time and consistency constraints in the system.

Definition 4 The miss rate, denoted by MR , is defined as N_{miss}/N_{total} , where N_{miss} and N_{total} are the number of mobile real-time read-only transactions missing their deadlines and the total number of transactions processed in the system respectively.

In this paper we follow the assumptions in [1] that the periodic broadcast is adopted, that is, all data items in the database have to be broadcast at least once in each broadcast cycle. Under periodic broadcast, if the waiting time is not of great importance, transactions need not issue explicit data request since all data items definitely appear in the broadcast sooner or later.

Definition 5 Let d^i be the i^{th} data item instance in a broadcast cycle and $\pi(i)$ is the time instance broadcasting d^i . If the following condition holds for d^i to d^{i+k-1} , we say that the k data item instances constitute a *consistency interval* $CI_m^{i,k}$, where i is the sequence number of the first item of $CI_m^{i,k}$ in the broadcast cycle, m is the monotonically increasing sequence number of $CI_m^{i,k}$ and k is the length of $CI_m^{i,k}$.

$$\forall d^j (i \leq j \leq i + k - 1) \wedge (S_{\pi(j)}(d^j) \in S_{\pi(i)}(DB))$$

The states of data items in $CI_m^{i,k}$ correspond to $S_{\pi(i)}(DB)$ and then are consistent with each other. Theoretically two successive consistency intervals may be overlapped and their lengths are not necessarily identical, while from the practical point of view, we only address *qualified* updates dissemination protocols with the following characteristics in this paper: 1) the lengths of consistency intervals are constant or nearly constant and 2) the end of one consistency interval is just the beginning of the next one. For simplicity of

presentation, in the following sections we use CI_m to represent $CI_m^{i,k}$ and C^k to denote a set of consistency intervals whose length is equal to k if there is no ambiguity.

3. BROADCAST PROTOCOLS

3.1 The System Model

The system model includes two relatively independent parts, the server-side model and the client-side model. The server-side model consists of the database server, the broadcast server and the mobile support station, which are interconnected by the fixed network. The database server maintains real-time data to be broadcasted, e.g., stock tickers, news updates and traffic conditions. We assume the size of each data item is identical and each update is labeled with a version number to indicate at which consistency interval the value is taken. Time in the system is measured in terms of the time duration for broadcasting a single data item.

The broadcast server delivers data through the downstream channel and supports two kinds of broadcast schemes, *Broadcast On-Schedule (BOS)* and *Broadcast On-Demand (BOD)*. *Broadcast On-Schedule* refers to all data items in the database are propagated according to the chosen scheduling algorithm such as flat broadcast or frequency-based scheduling strategy [6, 14, 16, 17], which aims to exploit the channel efficiently and decrease the access time as well as the tune in time. *Broadcast On-Demand* takes the user's explicit data requests coming from the upstream channel into consideration and delivers data items to meet their data requests. Both schemes may be incorporated into the updates dissemination protocols to ensure the consistency of transactions as well as achieve good performance in *SS* and *MR*. The mobile support station is responsible for managing the upstream channel, through which mobile clients send data requests if the desired data items do not appear timely. At the same time, a queue of data requests is maintained at the mobile support station, from which *Broadcast On-Demand* selects appropriate data items to deliver. Since the bandwidth for *Broadcast On-Demand* may be insufficient to meet all data requests, an ordering algorithm is necessary to prioritize data requests in the queue.

The client-side model is composed of a large number of mobile clients, from which mobile real-time read-only transactions are generated one at a time. Each *MRRT* consists of a set of ordered read operations and is associated with a deadline on its completion time. The term "ordered" means that a read operation can begin only after the preceding ones have finished. Each of read operations issued by an *MRRT* accesses only one data item by its key value. There are two ways for an *MRRT* to get its needed data items, from the broadcast as well as out of the local cache in the mobile client. The usage of cache can significantly reduce the waiting time of *MRRTs* while the cache consistency need to be validated periodically due to the updates occurred at the server. At the same time, the limited capacity of cache requires the cache replacement policy to be considered deliberately. If an *MRRT* cannot finish all read operations without violating the consistency criterion and time constraint, it has to be restarted as long as its deadline does not expire. The next run of the *MRRT* can read consistent data item from the local cache directly. Since the functions and behavior of transactions in real-time applications are much more

predictable [9, 11], we assume the number of read operations of an *MRRT* can be obtained using certain pre-analysis techniques before its execution.

3.2 Forward Multi-Version Data Broadcast

Updates dissemination protocols can utilize the consistency interval to ensure the transaction consistency. For example, if we choose the whole broadcast cycle as a consistency interval just like *MVB*, transactions that begin and commit in the same broadcast cycle are definitely consistent. However, the data currency is seriously affected since updates at the current broadcast cycle cannot be reflected in the data broadcast timely. Based on the transaction execution model described in section 3.1 and the definition of *SS*, we can get the following proposition easily.

Proposition 1 Given any *qualified* updates dissemination protocol, for two consistency intervals CI^{k_1} and CI^{k_2} , if $k_1 = n \times k_2$ where n is a positive number, *SS* of using CI^{k_2} is necessarily less than that of using CI^{k_1} .

With existing updates dissemination protocols, restarting transactions is the other common way to ensure the consistency. To continue their execution, restarted transactions have to wait until the new values of updated data items appear. However, the waiting time *depends on* particular scheduling algorithms and so is unpredictable, which is quite undesirable for transactions with deadline constraints. To reduce the uncertainty of the waiting time, a new policy is introduced for updates dissemination protocols in this paper.

Definition 6 Given any *qualified* updates dissemination protocol, suppose data item d is updated during consistency interval CI_m at the server and cannot be broadcast by the chosen scheduling algorithm during the remaining portion of CI_m , the *instantly broadcasting updates policy (IBUP)* delivers the new value of d immediately at the beginning of CI_{m+1} without waiting for being scheduled.

By the transaction execution model we can get proposition 2 readily.

Proposition 2 With a *qualified* updates dissemination protocol, using *IBUP* can provide the new values of updated data items to restarted transactions more timely than the one without *IBUP*.

Now we set out to introduce the preliminary version of our updates dissemination protocol – *forward multi-version data broadcast (FMVB)*. As discussed above, *MVB* seriously affects the data currency mainly due to its long broadcast cycle and delivering old versions of data items. Therefore, two improvements are made for *MVB* in the design of *FMVB*. The first improvement is shortening the length of consistency intervals as well as only delivering the latest versions of data items during each consistency interval. According to Proposition 1, appropriate length candidate may be $1/n$, e.g., half or one third, of the length of the whole broadcast cycle, where n is a natural number greater than 1.

This improvement can dramatically decrease the probability of reading stale data for *MRRTs*.

In terms of Proposition 2, the second improvement for *MVB* is utilizing *IBUP* to shorten the execution time of restarted *MRRTs*. In particular, at the beginning of each consistency interval the *new values (NV)* of updated data during the preceding consistency interval are delivered immediately instead of waiting for being broadcast by the chosen scheduling algorithm. Due to the usage of *IBUP*, there may be multiple versions of one data item in each broadcast cycle. But unlike *MVB*, the number of versions is not fixed and the latest version always goes forward with the execution of update transactions at the server. That is just why we call it *forward multi-version data broadcast*. To maintain the consistency, at the beginning of each consistency interval (just before *NV*) an *invalidation report (IR)* is propagated similar to *IM*, in which there are the identifiers of data items updated during the preceding consistency interval. Mobile real-time read-only transaction, say *T*, reads *IR* periodically and determines to restart or not by judging whether data items in its read set are found in *IR*. If an inconsistent read occurs, *T* has to restart to ensure the consistency.

The broadcast organizations of *MVB*, *IM*, *FMVB* and *HFMVB* (discussed in the later subsection) are compared in Fig. 1.

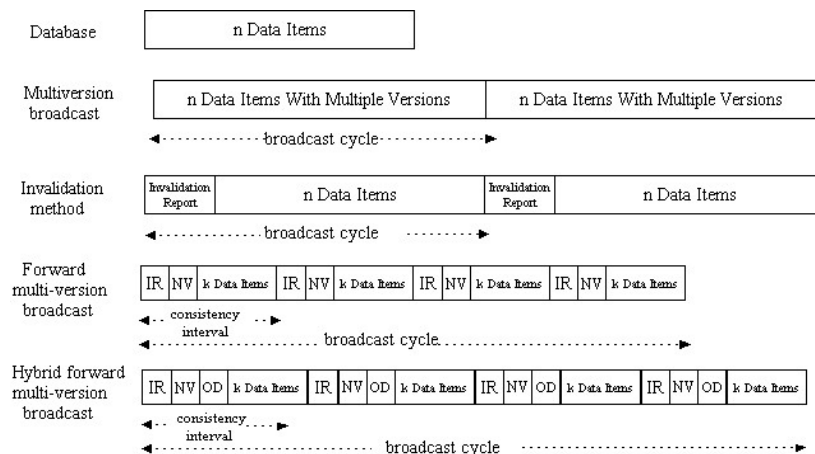


Fig. 1. The broadcast organizations of *MVB*, *IM*, *FMVB* and *HFMVB*.

As discussed in section 3.1, mobile clients may cache data items of interest locally. In the presence of updates, items in the cache may become stale. There are various approaches to communicating updates to the client caches. Invalidation combined with a form of autoprofing was shown to perform well in broadcast delivery [4]. Similar to [4], in *FMVB* this kind of cache updates is used. In particular, mobile clients tune in at the beginning of each consistency interval and then read *IR*. If data items in cache are found in *IR*, they are set as invalidated and remain in cache. When the new value of an invalid-dated data item appears in the broadcast, the client fetches the new value and replaces the old one once they appear in *NV*. Thus, a data item in cache either has a cur-

rent value or is marked as invalidated. For an *MRRT*, if all needed data items are available in cache as well as not invalidated, it can commit at once. Otherwise it must wait until desired data items are broadcast. As to the cache replacement policy, the most commonly used one, *LRU*, is adopted in this paper.

The detailed server-side protocol of *FMVB* is depicted in Fig. 4 only excluding line 8, and client-side protocol of *FMVB* is depicted in Fig. 3 excluding lines 5-11 and line 22.

3.3 Hybrid Forward Multi-Version Data Broadcast

In the context of data broadcast, an *MRRT* can safely commit as long as the desired data items consistently appear before its deadline. However, since the underlying notion of broadcast is to disseminate the whole database to all *MRRTs*, certain *MRRTs*, especially those with short deadlines, will inevitably violate their time constraints due to the long broadcast cycle. To solve the problem, we will address a further version of *FMVB*, *hybrid forward multi-version data broadcast (HFMVB)*, by combining *FMVB* with *Broadcast On-Demand* technique in this subsection.

Results presented in [13] indicate that *Broadcast On-Demand*, which makes use of the low bandwidth upstream channel to accept data requests and then delivers data on-demand to meet data needs with deadline constraints, has a better performance compared with pure data broadcast (*Broadcast On-Schedule*). Similar research work in [2, 3, 6] also shows that hybrid data broadcast, which joins *Broadcast On-Schedule* with *Broadcast On-Demand*, performs better than the cases just using a single broadcast policy. While *FMVB* in conjunction with *Broadcast On-Demand* seems to be promising in reducing *MR*, several technical problems need to be resolved first.

The first one is how to allocate limited bandwidth of the downstream channel between *Broadcast On-Schedule* and *Broadcast On-Demand*. If the bandwidth for *BOD* is too small, only a few *MRRTs* can meet the timelines thanks to *BOD*. In contrast, if we allocate less bandwidth for *BOS*, the overall length of broadcast cycle will increase (Recall that all data items must be delivered at least once by the scheduling algorithm through the bandwidth for *BOS*), which may result in an increasing amount of data requests and overload the server. Since all requested data items need to be broadcast by *BOD*, thus if *BOD* cannot deal with overmuch data requests in time, negative impact on the miss rate will occur due to the latency in processing data requests. Considering the complexity of bandwidth allocation, the ratio of *BOD* to *BOS* is fixed in *HFMVB* and may be adjusted as a system parameter in practice. With respect to the period of *BOD*, it is set the same as that of *CI* to simplify the implementation as well as provide much higher data currency. The organization of *HFMVB* is illustrated in Fig. 1.

The second problem is how to determine the time when *MRRTs* send their data requests if necessary. Although this problem is either ignored or simplified in previous literature, e.g., the intersection of contents of *BOD* and *BOS* is assumed empty [13], it is very important here owing to the following reason. As mentioned above, transactions can acquire data items of interest sooner or later thanks to the periodic data broadcast, but it is just valuable for an *MRRT* to see all needed data items consistently before its deadline. Thus, from this angle, *MRRTs* should send data requests immediately once the desired data items are not available on the channel. However, the upstream channel is usually restricted in the context of broadcast and too many data requests may lead to the channel

contest as well as swamp the server, which in turn result in long latency from a data request is initiated to the server processes it. In addition, sending messages is an energy consumption operation and the reduction of unnecessary data requests will conserve the limited energy of mobile clients [3, 18]. Therefore, it is very important to provide precise information for *MRRTs* to judge whether it is necessary to issue a data request on seeing the desired data item is not available on the channel.

Index techniques are widely used in broadcast organization for mobile clients to locate data of interest precisely [3, 8, 10] and then may be a good candidate to solve the second problem. However, existing index techniques assume that the contents of data broadcast is known a priori and require indexes must be constructed in advance, which is not suitable for *HFMVB* due to the dynamic change in contents of *NV* and *OD* with the execution of update transactions and data request processing at the server. Thus, in the following section we will adapt existing index techniques to the requirements of *HFMVB*.

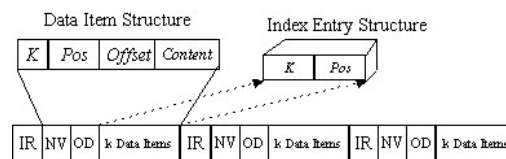


Fig. 2. Index entry structure and data item structure in *HFMVB*.

Similar to [10], we assume a B-tree structure for the index. As depicted in Fig. 4, for each broadcast cycle, only the contents of *BOS*, i.e., all data items excluding *NV* and *OD*, is used to construct the index. Because the chosen scheduling algorithm produces predictable broadcast schedule, the difficulty and complexity of index dynamic generation can be avoided. The whole index consists of a series of index entries and other special attached information. Each index entry consists of a 2-tuple $\langle K, Pos \rangle$, where *K* is the key value of corresponding data item; *Pos* is the relative position of the data item in *BOS*. Each data item in data broadcast consists of a 4-tuple $\langle K, Pos, Offset, Content \rangle$, where *K* is the key value of the data item; for data items not within *NV* and *OD*, *Pos* is equal to its relative position in the broadcast schedule and *Offset* is equal to 0; for data items within *NV* and *OD*, *Pos* is equal to the *Pos* value of the last data item in the preceding consistency interval and *Offset* is equal to its relative position in *NV* and *OD*; *Content* is composed of the other fields of the data item.

To determine the length of *NV* and *OD*, the estimation values of the update rate (defined as the number of data items updated per broadcast cycle divided by the whole database size) at the server, denoted by *UpdateRate*, and the bandwidth ratio of *BOD* to *BOS*, denoted by *Ratio*, are included in the index, where *UpdateRate* can be the average of update rates in last several broadcast cycles or other meaningful estimations in terms of practical application scenarios. In addition, the size of database and the length of *BOS*, denoted by *N* and *SCH(N)* respectively, are also attached in the index. Based on the above index structure and attached information, for any read operation issued by *MRRTs*, say $r(d)$, the time interval from the read operation begins to the instant at which the desired data items appear can be got by Eq. (1) (Note that time is measured in terms of the time duration for broadcasting a single data item).

$$NODP(d) = (IPos - Pos + Offset) + (IPos \bmod k - Pos \bmod k) \times ((SCH(N) \times Ratio + N \times UpdateRate)/m) \quad (1)$$

where Pos and $Offset$ are the corresponding values of the data item being broadcasted when the read operation begins; $IPos$ is the Pos value of the needed data item got by index searching; **mod** is the modulus operation; m is the number of consistency intervals per broadcast cycle, i.e., $m = SCH(N)/k$. In Eq. (1), $(IPos - Pos + Offset)$ refers to the distance between the current broadcasting item and the desired item in BOS and $Offset$ is used to revise this equation if the current broadcasting item belongs to NV and OD . $(IPos \bmod k - Pos \bmod k)$ denotes how many NV and OD exist between the current broadcasting item and the desired item. $(SCH(N) \times Ratio + N \times UpdateRate)/m$ is the size of NV and OD per consistency interval.

By comparing $NODP(d)$ and d 's deadline one can know whether the needed data item may appear in time. However, $MRRT$ s often have more than one ordered read operations. Thus, the transaction deadline should be assigned to all data items of interest. In $HFMVB$ we use the serial assignment policy, i.e., dividing the remaining execution time of an $MRRT$ equally into all pending read operations. The deadline of each data item can be determined according to Eq. (2).

$$DL(d_i) = (Deadline - CurrentTime)/(NumQueries - i + 1) \quad (2)$$

where $DL(d_i)$ is deadline of the i^{th} data item that an $MRRT$ needs; $Deadline$ is the deadline of the $MRRT$; $CurrentTime$ is the time now; $NumQueries$ is the number of read operations of the $MRRT$. Note that the deadline of each pending data item needs to be recalculated once the current read operation is met.

When an $MRRT$ initiates a read operation, say $r(d_i)$, if $DL(d_i)$ is greater than $NODP(d_i)$ the transaction will wait. Otherwise, a data request along with its data deadline will be generated. In the broadcast server the data requests are queued and processed according to EDF scheduling policy, that is, data requests with shorter data deadlines will be processed with higher priority [12]. The detailed implementation of $HFMVB$ protocol is depicted in Figs. 3 and 4.

4. PERFORMANCE EVALUATION

4.1 Simulation Model

Our simulation model is similar to the one presented in [4]. There are $NumItems$ data items in the database. The bandwidth proportion of BOD to BOS is $Ratio$. Index is broadcast $NumIndexes$ times per broadcast cycle. Update transactions at the server are generated per $UpdateInterval$ and their update operations follow a Zipf distribution with parameter θ_u . The update distribution is across the range 1 to $UpdateRange$. The length of consistency intervals is $ConsistencyInterval$. The basic time unit in our simulation is the time used to broadcast a data item.

```

Procedure DataAcquire( $d_i$ ,  $Read\_Set$ )
Input:  $d_i$  is the  $i^{\text{th}}$  data item needed by the transaction in execution
         $Read\_Set$  is the read set of the transaction in execution
1: if ( $d_i$  is in local cache) then
2:   Add  $d_i$  into  $Read\_Set$  of the transaction that issues the read operation;
3: else
4:   Listen to the data broadcast;
5:   Read current broadcasted data item and get its ( $Pos$ ,  $Offset$ );
6:   Search the index and locate  $d_i$ 's relative position  $IPos$ ;
7:    $DL(d_i) = (Deadline - Current\_Time)/(NumQueries - i + 1)$ ;
8:    $NODP(d_i) = (IPos - Pos + Offset) + (IPos \bmod k - Pos \bmod k) \times ((SCH(N) \times Ratio + N \times UpdateRate)/m)$ ;
9:   if ( $NODP(d_i) > DL(d_i)$ ) then
10:    Send a data request to the broadcast server;
11:   else
12:     if (it is time to receive  $IR$ ) then
13:       if (data items in  $Read\_Set$  are found in  $IR$ ) then
14:         Restart the transaction and re-read data items from local cache and  $NV$ ;
15:       endif
16:     else
17:       if ( $d_i$  appears) then
18:         Read  $d_i$  and add it into  $Read\_Set$ ;
19:       endif
20:     endif
21:   endif
22: endif

```

Fig. 3. Client side protocol of *HFMVB*.

```

Procedure GeneratingBroadcastProgram( $Update\_Set$ ,  $k$ )
Input :  $Update\_Set$  is the set of identifiers of data updated in the preceding consistency interval
         $k$  is the length of consistency intervals of the given updates dissemination protocol
1: while (true)
   /*Each while-loop accomplishes the data dissemination of one consistency interval*/
2:    $Temp = Update\_Set$ ;
3:    $Update\_Set = \phi$ ;
4:   if ( $Temp \neq \phi$ ) then
5:     broadcast invalidation report in terms of data items in  $Temp$ ;
6:     broadcast the new values of data items in  $Temp$ ;
7:   endif
8:   broadcast  $k \times Ratio$  data items according to the data requests queue and EDF scheduling policy;
9:   broadcast  $k$  data items by the chosen broadcast scheduling algorithm;
10: endwhile

```

Fig. 4. Server side protocol of *HFMVB*.

The client simulator initiates an *MRRT* per *ThinkInterval*. Each *MRRT* has *NumQueries* read operations. The deadline of *MRRT* is equal to $CurrentTime + V \times NoQueries$, where V is a value randomly selected between *MinLaxity* and *MaxLaxity*. The read operations of *MRRTs* access data items from the range 1 to *ReadRange*. The access probabilities follow a Zipf distribution with a parameter θ . The cache size in the client is *CacheSize* and cache replacement policy is *LRU*. When pages are updated, the corresponding cache entries are invalidated and subsequently autoperfetched. The performance metrics are the miss rate and the system staleness. The main parameters and the baseline values are summarized in Table 1.

Table 1. Parameters and baselines.

Server Parameters		Client Parameters	
NumItems	1000	ReadRange	1000
UpdateRange	1000	ThinkInterval	4
UpdateInterval	40	NumQueries	4
ConsistencyInterval	200	MinLaxity	500
Ratio	0.1	MaxLaxity	1000
NumIndexes	5	CacheSize	125
θ_u	0.9	θ_r	0.9

4.2 Results and Discussion

Experiment 1: We change *UpdateInterval* to compare the performance metrics of *MVB*, *IM*, *FMVB* and *HFMVB* under different update rate. The number of versions for *MVB* is 2.

The miss rates of these four protocols are compared in Fig. 5. *IM* has the highest one due to its highest restart rate and the lack of *IBUP*. Although the restart rate of *FMVB* is higher than those of the others, the shortened waiting time of restarted transactions thanks to *IBUP* leads to its better performance than *IM*. Compared with *MVB*, the miss rate of *FMVB* is higher due to its high restart rate when the update rate is high. While with the decrease of the update rate, the difference of the miss rates between *FMVB* and *MVB* diminishes gradually. Since *HFMVB* can meet more transactions with short deadlines by hybrid broadcast policy, it performs best.

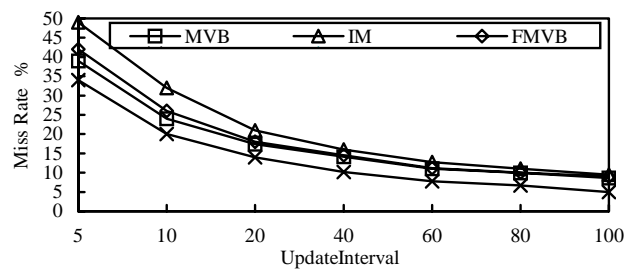


Fig. 5. The miss rate comparison.

The system staleness of *MVB*, *IM* and *FMVB* are depicted in Fig. 6. This metric of *HFMVB* is not given since it is similar to that of *FMVB*. The system staleness of all three protocols decreases with the reduction of the update rate. *MVB* has the highest one because it allows transactions to read stale versions of data. Owing to the invalidation method, the metrics of *IM* and *FMVB* are observably better than that of *MVB*. Compared with *IM*, *FMVB* has a better performance since the shorter consistency interval further decreases the probability of reading outdated data. Because transactions begin and commit during one consistency interval still may read outdated data, the system staleness of *FMVB* is not equal to 0.

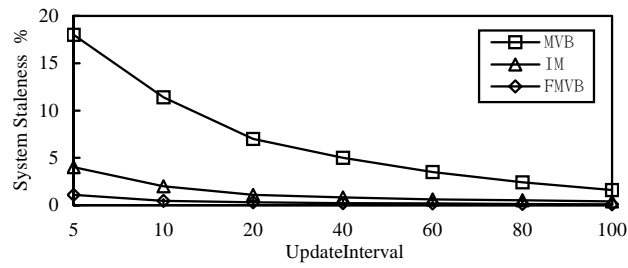


Fig. 6. The system staleness comparison.

Experiment 2: In this experiment we change the system workload and the bandwidth ratio of *BOD* to *BOS* to observe the variation of performance metrics for *FMVB* and *HFMVB*.

In Fig. 7 the miss rates of *FMVB* and *HFMVB* are illustrated under different values of *ThinkInterval*. As *ThinkInterval* is across the range from 8 to 0.0625, the amount of data needed by *MVRTs* changes from 0.5 to 6 times as many as the amount of data broadcast, i.e., the system workload increases by degrees. From Fig. 7 we can see that with the increase of the system workload, the miss rate of *HFMVB* increases accordingly due to the limited bandwidth of broadcast on-demand. But it does not exceed that of *FMVB* under the same update rate.

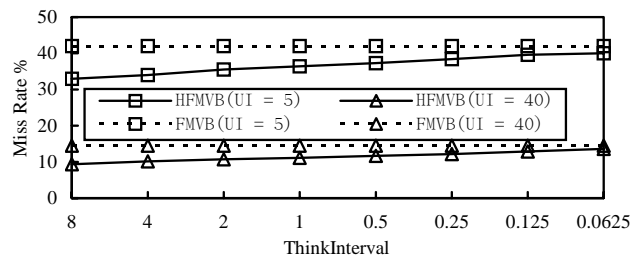


Fig. 7. The miss rate comparison.

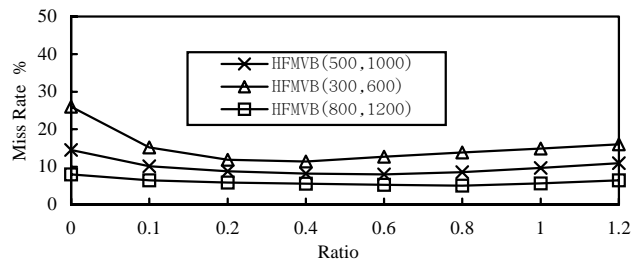


Fig. 8. The miss rate comparison.

Fig. 8 illustrates the miss rate of *HFMVB* with the variation of *Ratio* under different transaction laxities. From Fig. 8 we can see that the miss rate does not constantly de-

crease with the increase of *BOD* bandwidth but ascends slightly after Ratio exceeds one critical point. Reason for this phenomenon may be that the increase of bandwidth for *BOD* results in the longer broadcast cycle, which in turn leads to transactions that can get their needed data through *BOS* before now have to acquire data from *BOD*. When the increasing speed of data requests becomes larger than that of bandwidth for *BOD*, the miss rate may drop instead of constantly increasing. At the same time, the transaction laxities also have an impact on the critical point. The shorter the transaction laxities are, the earlier the critical point emerges.

5. CONCLUSION AND FUTURE WORK

In this paper, we study the problem of providing consistent data to mobile read-only real-time transactions in data broadcast environment. The main contributions of this paper include the following. First, we formally define the performance objectives based on the detailed requirement analysis of update dissemination protocols for mobile real-time read-only transaction processing. Next, a new updates dissemination protocol called *hybrid forward multi-version data broadcast* is proposed and the correctness of *HFMVB* is validated. Finally, extensive simulation experiments are implemented and numerical results show that *HFMVB* has better performance compared with existing updates dissemination protocols and is more appropriate for mobile read-only real-time transaction processing.

REFERENCES

1. S. Acharya, M. Franklin, and S. Zdonik, "Disseminating updates on broadcast disks," in *Proceedings of the 22nd Very Large Data Base Conference*, 1996, pp. 354-365.
2. S. Acharya, M. Franklin, and S. Zdonik, "Balancing push and pull for data broadcast," in *Proceedings of ACM SIGMOD International Conference on Management of Data*, 1997, pp. 183-194.
3. A. Datta, A. Celik, J. Kim, and D. E. VanderMeer, "Adaptive broadcast protocols to support power conservant retrieval by mobile users," in *Proceedings of International Conference on Data Engineering*, 1997, pp. 124-133.
4. E. Pitoura and P. K. Chrysanthis, "Multiversion data broadcast," *IEEE Transactions on Computers*, Vol. 51, 2002, pp. 1224-1230.
5. E. Pitoura and P. K. Chrysanthis, "Exploiting versions for handling updates in broadcast disks," in *Proceedings of Very Large Data Base Conference*, 1999, pp. 114-125.
6. J. Fernandez and K. Ramamritham, "Adaptive dissemination of data in real-time asymmetric communication environments," in *Proceedings of Euromicro Conference on Real-Time Systems*, 1999, pp. 195-203.
7. J. Shanmugasundaram, A. Nithrakashyap, R. Sivasankaran, and K. Ramamritham, "Efficient concurrency control for broadcast environments," in *Proceedings of ACM SIGMOD International Conference on Management of Data*, 1999, pp. 85-96.
8. Q. Hu, D. L. Lee, and W. C. Lee, "A comparison of indexing methods for data

- broadcast on the air,” in *Proceedings of the 12th International Conference on Information Networking*, 1998, pp. 656-659.
9. K. Y. Lam, T. W. Kuo, W. H. Tsang, and C. K. Law, “Concurrency control in mobile distributed real-time database systems,” *Information Systems*, Vol. 25, 2000, pp. 261-286.
 10. T. Imielinski, S. Viswanathan, and B. R. Badrinath, “Data on air: organization and access,” *IEEE TKDE*, Vol. 9, 1994, pp. 353-372.
 11. E. Kayan and O. Ulusoy, “Real-time transaction management in mobile computing systems,” in *Proceedings of the 6th International Conference on Database Systems for Advanced Applications*, 1999, pp. 127-134.
 12. T. Bowen, G. Gopal, G. Herman, T. Hickey, K. Lee, W. Mansfield, J. Raitz, and A. Weinrib, “The datacycle architecture,” *Communications of the ACM*, Vol. 35, 1992, pp. 71-81.
 13. P. Xuan, O. Gonzalez, J. Fernandez, and K. Ramamritham, “Broadcast on demand: efficient and timely dissemination of data in mobile environments,” in *Proceedings of the 3rd IEEE Real-Time Technology Application Symposium*, 1997, pp. 38-48.
 14. G. L. Lee, S. C. Lo, and A. L. P. Chen, “Data allocation on wireless broadcast channels for efficient query processing,” *IEEE Transactions on Computers*, Vol. 51, 2002, pp. 1237-1251.
 15. R. Rastogi, S. Mehrotra, Y. Breitbart, H. F. Korth, and A. Silberschatz, “On correctness of non-serializable executions,” in *Proceedings of ACM Symposium on Principles of Database Systems*, 1993, pp. 97-108.
 16. H. V. Leong and A. Si, “Data caching over the air-storage,” *The Computer Journal*, Vol. 40, 1997, pp. 401-415.
 17. C. J. Su and L. Tassiulas, “Broadcast scheduling for information distribution,” in *Proceedings of the IEEE INFOCOM*, 1997, pp. 109-117.
 18. D. Darbara, “Mobile computing and database: a survey,” *IEEE Transactions on Knowledge and Data Engineering*, Vol. 11, 1999, pp. 108-117.



Hong-Ya Wang (王洪亚) received his B.S. degree in Education Information Technology and M.S. degree in Electrical engineering from Central China Normal University, P.R.C., in 1998 and 2001, respectively. In 2005 he received his Ph.D. degree in Computer Science in Huazhong University of Science and Technology. He is now a lecturer at School of Computer Science and Technology, Donghua University. His main research interests include mobile computing, real-time computing, data stream processing, and advanced database systems.



Guo-Qin Ning (宁国勤) is a Ph.D. candidate in Information and Communication Engineering at Huazhong University of Science and Technology. His research interests are in the areas of various wireless access technologies, wireless multimedia communications, and mobile computing. He received his B.S. degree in Education Information Technology and M.S. degree in Computer Application from the Central China Normal University in 1998 and 2002, respectively. He is also a lecturer at Department of Information Technology, Central China Normal University.



Guo-Hui Li (李国徽) received his Ph.D. in Computer Science in 1999 from Huazhong University of Science and Technology. Currently he is a Professor in Huazhong University of Science and Technology. His main research interests are mobile computing, real-time computing and advanced databases, including real-time database, active database and main memory database.



Kam-Yiu Lam (林金耀) received the B.Sc. (Hons) degree in Computer Studies with distinction and Ph.D. degree from the City University of Hong Kong in 1990 and 1994, respectively. He is currently an Associate Professor in the Department of Computer Science, City University of Hong Kong. His current research interests are real-time database systems, real-time active database systems, mobile computing, and distributed multimedia systems. Dr. Lam is a member of the IEEE.