Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®
The Journal of Systems and Software 69 (2004) 183–193
The Journal of Systems and Software
www.elsevier.com/locate/jss

Efficient validation of mobile transactions in wireless environments ☆

Victor C.S. Lee [a,*], Kwok Wa Lam [a], Tei-Wei Kuo [b]

[a] Department of Computer Science, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon, Hong Kong
[b] Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan

Received 30 September 2001; received in revised form 31 January 2002; accepted 19 July 2002

Abstract

In broadcast environments, the limited bandwidth of the upstream communication channel from the mobile clients to the server bars the application of conventional concurrency control protocols. In this paper, we propose a new variant of the optimistic concurrency control (OCC) protocol that is suitable for broadcast environments. At the server, forward validation of a transaction is done against currently running transactions, including mobile transactions and server transactions. At the mobile clients, partial backward validation of a transaction is done against committed transactions at the beginning of every broadcast cycle. Upon completion of execution, read-only mobile transactions can be validated and committed locally and update mobile transactions are sent to the server for final validation. These update transactions have a better chance of commitment because they have gone through the partial backward validation. In addition to the nice properties of conventional OCC protocols, this protocol provides autonomy between the mobile clients and the server with minimum upstream communication, which is a desirable feature to the scalability of applications running in broadcast environments. This protocol is able to process both update transactions and read-only transactions at the mobile clients at low space and processing overheads.

Keywords: Real-time transaction processing; Optimistic concurrency control; Transaction validation; Mobile computing

1. Introduction

In near future, tens of millions of users will be carrying a portable computer that uses a wireless interface to access the worldwide information network for business or personal use (Imielinski and Badrinath, 1994). However, some physical constraints of wireless communication pose a number of challenging issues on transaction processing. In a wireless mobile network, the server may have a relatively high downstream bandwidth broadcast capability while the upstream bandwidth for mobile clients to the server is very limited. Such asymmetric communication environments (Acharya et al., 1995) render the conventional transaction processing mechanisms inapplicable because those mechanisms require considerable bi-directional communication between the server and mobile clients and the time required for message passing may be intolerably long. Another issue is that data transmission over the air is monetarily expensive as the bandwidth from mobile clients to the server will continue to be a scarce resource (Pitoura and Bhargava, 1994). The prolonged communication time may be too costly. Furthermore, the large population size of mobile clients may overload the server when they asynchronously submit transactions to the server for processing. Therefore, one of the design objectives of our protocol is to minimize the use of upstream bandwidth.

To exploit the relative abundance of downstream communication capacity from the server to the mobile clients, broadcast-based data dissemination becomes a major mode of information transfer in mobile computing and wireless environments (Alonso and Korth, 1993; Imielinski and Badrinath, 1994; Zdonik et al., 1994). Therefore, we assume such broadcast mechanism in our

☆ The work described in this paper was partially supported by a grant from CityU (project no. 7001193) and a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (project no. CityU 1075/02E). This paper is a complete and extended version of a paper containing the preliminary idea in the First International Conference on Mobile Data Access, 1999.

* Corresponding author. Tel.: +852-2788-8617; fax: +852-2788-8614.
E-mail address: csvlee@cityu.edu.hk (V.C.S. Lee).

0164-1212/$ - see front matter © 2003 Elsevier Inc. All rights reserved.
doi:10.1016/S0164-1212(03)00084-0

proposed protocol. Broadcast disks (Acharya et al., 1995) are a form of such data dissemination systems. The server continuously and repeatedly broadcasts all data objects in the database. The mobile clients view this broadcast as a disk and read the values of data objects being broadcast. A periodic broadcast program is constructed to schedule the broadcast of data objects cyclically according to certain popularity criteria. Some unused extra broadcast slots in each broadcast cycle can be used to broadcast additional information such as control information to be used by the mobile clients to perform some functions on their local transactions.

Many applications in broadcast environments are inherently real-time in nature. Transactions are associated with timing constraints in form of deadlines, independent of whether they originate from the mobile clients or the static hosts over wired or wireless networks. For instance, it may be a financial or opportunity loss if a stock-trading transaction cannot be completed with a certain timing constraint, disregarding whether the stock trader is submitting the transaction (purchasing or read-only) in his office (wired) or on a ride to somewhere (wireless). In addition, the temporal validity of some data objects such as stock prices or sensor data poses another type of timing constraints to the database systems. Transaction correctness is then defined as meeting its timing constraints and using data that is absolutely and relatively timing consistent (Stankovic and Zhao, 1988; Stankovic et al., 1999).

Most of the systems and applications in mobile computing environments comprise of a large proportion of read-only transactions. Read-only transactions do not modify any data (Garcia-Molina and Wiederhold, 1982). Examples include information dispersal systems for temporal or time-sensitive information such as stock prices, traffic condition and weather information. In electronic commerce applications, such as stock markets and auctions, it is expected that the number of stock buyers or bidders is relatively few compared to the number of speculators who look up (read) the prices frequently. The large population of read-only transactions makes the processing of read-only transactions an important performance issue in these applications (Kuo et al., 1998). Although, read-only transactions can be processed with conventional transaction processing algorithms, in many cases it is more efficient to process read-only transactions with special algorithms which take advantage of the knowledge that the transactions only read (Garcia-Molina and Wiederhold, 1982; Lam et al., 1998). Therefore, another design objective of our proposed protocol is to exploit the semantics of read-only transactions at mobile clients such that they can be processed separately from the update transactions at the server.

The rest of this paper is organized as follows. Section 2 discusses some related work. Section 3 discusses important issues of transaction processing in broadcast environments that induce the development of our proposed protocol. Our proposed protocol will be described in Section 4. Section 5 shows the performance of our proposed protocol compared with the conventional optimistic concurrency control (OCC) protocol using simulation. We conclude the study and discuss any enhancement and optimization that can be done in near future in Section 6.

2. Related work

Data management in wireless environments receives a lot of attention in these few years (Barbara, 1997; Dunham et al., 1997; Huang et al., 1994; Imielinski and Badrinath, 1994; Lam et al., 1999; Lee et al., 1998; Pitoura and Bhargava, 1994; Pitoura and Samaras, 1998; Shanmugasundaram et al., 1997). However, there are few studies on transaction processing in wireless environments, and nearly all of them are focused on processing of read-only transactions. In particular, a data-cycle architecture (Herman et al., 1987) is introduced for high throughput database systems. The entire contents of the database are repetitively broadcast to the hosts in a high bandwidth network. Shanmugasundaram et al. (1999) showed that it is very expensive to use serializability as correctness criterion in the data-cycle architecture. In his pioneer work (Shanmugasundaram et al., 1999), a correctness criterion is proposed to allow read-only transactions to read current and consistent data in wireless environments without contacting the server. However, serializability is not maintained in their protocol. Two different read-only transactions may perceive the effects of update transactions in different serialization orders. It may be hazardous to certain applications such as mobile stock trading where a buy/sell trade will be triggered to exploit the temporary pricing relationships among stocks. From the trader's perspective, the inability of database management system to maintain the serializability may have important financial consequences (Bowen et al., 1992). For instance, if the users who submit multiple read-only transactions communicate and compare their query results, they may be confused (Garcia-Molina and Wiederhold, 1982). Distinct from the past work, the protocol proposed in this paper considers both read-only and update transactions at the mobile clients, and the proposed protocol maintains the serializability at low cost in terms of a smaller control information size.

In another comprehensive work (Pitoura, 1998; Pitoura and Chrysanthis, 1999), a number of broadcast methods are introduced to guarantee the correctness of read-only transactions. The *multiversion broadcast* approach broadcasts a number of versions for each data item along with the version numbers. This method

increases the size of the broadcast cycle and accordingly the response time. Moreover, the serialization order is fixed at the beginning of the read-only transaction. It is usually too restrictive and lacks flexibility. In the *invalidation-only broadcast*, a read-only transaction is aborted if any data item that has been read by the read-only transaction is updated at the server, and it results in low concurrency. For the *conflict-serializability* method, both the mobile clients and the server have to maintain a copy of the serialization graph for conflict checking. It incurs high overheads to maintain the serialization graph. The integration of updates into the local copy of the serialization graph and the cycle detection may be too computation intensive for portable mobile computers.

Barbara (1997) used a modified version of OCC to support both read-only and update transactions at the mobile client side. However, in their protocol, they adopted a certifier that aborts mobile transactions whenever there are data conflicts with server transactions. This mechanism will cause intolerable delay to mobile transactions that are aborted when they are submitted to the server for completing the verification. In addition, read-only transactions are not discriminately handled in their protocol. On the other hand, in our proposed OCC variant, we adopt forward validation at the server such that mobile transactions that are submitted to the server for final validation are more likely to commit. Therefore, fewer mobile transactions will be restarted at the server and more processing and communication resources will be saved. Furthermore, read-only transactions are handled separately in our protocol such that they are allowed to commit locally at the mobile client side without contacting the server. As a result, the system performance is further enhanced.

3. Issues of transaction processing in broadcast environments

In this section, some important issues of transaction processing in real-time broadcast environments are discussed. These issues are then addressed in the design of our proposed protocol.

3.1. Asymmetric communication and limited capability of mobile computers

The first most critical constraint in wireless environments is the limited upstream bandwidth from the mobile clients to the server. So, a desirable concurrency control protocol should minimize the use of upstream communication channel. Other constraints are the limited capability and battery life of the mobile clients. It is of particular importance to reduce the amount of unproductive processing in mobile computers.

Conventional concurrency control protocols are based on locking and optimistic approaches. However, locking-based protocols are not feasible in wireless environments (Shanmugasundaram et al., 1999) because lock requests for each data object from the mobile clients to the server require excessive bi-directional communication that is too expensive in light of the limited upstream bandwidth. The transaction response time will be intolerably long due to the low bandwidth in wireless environments. Furthermore, the population size in broadcast environments is usually very large. A huge number of lock requests will overload the server.

On the other hand, OCC protocols are preferable. In OCC mechanism, transactions are allowed to execute unhindered until they reach their commit point, at which time they are validated (Kung and Robinson, 1981). This approach suits the asymmetric communication bandwidth property in broadcast environments. The server can use the large downstream bandwidth to broadcast data objects to transactions at the mobile clients where they can be processed locally without sending every request to the server. After a transaction finishes reading and pre-writing all the requested data objects at the mobile client, the transaction together with the required information is sent back to the server for validation. This basic and straightforward extension of optimistic approach helps to relieve the limited upstream communication bandwidth. However, this approach suffers from a number of overheads. Firstly, a large number of validation requests may overload the server and the server has to keep a long historical record of committed transactions for validation. Secondly, some mobile transactions, which are destined to restart due to data conflicts with committed transactions at the server, are allowed to execute to the end and are sent to the server for validation. Processing of these destined-to-restart transactions is useless and wastes the limited upstream communication bandwidth. Eventually, these overheads may lead to insufficient time for mobile transactions to complete their execution before deadlines. To solve these problems, our proposed OCC variant can detect data conflicts and restart the transactions at early stage of execution at the mobile client side. As a result, unfruitful processing and communication can be eliminated. Moreover, transactions may have greater chance to complete before their deadlines after restart.

3.2. Serializability

Due to the asymmetric communication bandwidth between the mobile clients and the server, existing concurrency control protocols for transaction processing are not suitable for broadcast environments. Hence, recent work (Shanmugasundaram et al., 1997; Shanmugasundaram et al., 1999) relaxed the strictness of

serializability and proposed some concurrency control protocols based on the relaxed consistency requirements for broadcast environments. While these protocols are useful in some real-time applications, serializability is needed to guarantee correctness of some applications in real-time broadcast environments.

Serializability is the standard notion of correctness in transaction processing (Bernstein et al., 1987) to preserve database consistency. Serializability requires that concurrent transactions be scheduled in a serializable way, i.e., the result produced by the interleaved execution of a set of transactions should be equal to the result produced by executing the transactions in some serial order. When transactions are processed in a serializable manner, the database is guaranteed to remain in a consistent state. Therefore, the objectives of concurrency control in transaction processing are to maintain the serializability on the one hand and to maximize concurrency on the other so that transactions are more likely to complete within their deadlines (Peng and Lin, 1996; Yu et al., 1994). To illustrate the importance of the serializability in transaction processing in wireless environments, let us take mobile stock trading as an example. Consider a stock-trading read-only transaction Q_1 at a mobile client that reads the number of units (X) and the unit price (Y) of a stock. Let U_2 and U_3 be update transactions at the server that update the values of X and Y, respectively.

Consider the following execution schedule:

$$r_1(X)r_2(Y)r_2(X)w_2(X)c_2 \quad r_3(Y)w_3(Y)c_3r_1(Y)\cdots$$

If Q_1 was allowed to commit, then both the server and the mobile client would see serializable executions. The serialization order for the server is $U_2 \rightarrow U_3$ whereas the serialization order for Q_1 at the mobile client is $U_3 \rightarrow Q_1$ ($\rightarrow U_2$). Assume that the execution schedule is allowed, and the number of units X of the stock is reduced from 1100 to 1000 and is updated by U_2 whereas the unit price Y of the stock drops to \$0.9 and is updated by U_3. Since Q_1 perceives a different serialization order from that by the server, it would appear to Q_1 that there were 1100 units of the stock with unit price of \$0.9. This is confusing. In fact, the global execution schedule ($Q_1 \rightarrow U_2 \rightarrow U_3 \rightarrow Q_1$) is not serializable. In order to avoid a transaction to see a logically inconsistent state (Bober and Carey, 1992), the above execution schedule should not be allowed.

3.3. Control information

To guarantee the serializability, additional control information has to be broadcast by the server to the mobile clients. The size and the complexity of control information affect the performance of the system. First of all, sending control information consumes bandwidth. If the size of control information is comparable to the size of database to be broadcast, it will be inefficient in terms of bandwidth utilization. Secondly, the size of control information attributes to the length of broadcast cycle, which in turn has a great impact on the satisfaction of timing constraints of transactions at the mobile clients. There are two main components that affect transactions meeting their deadlines at the mobile clients. The first one is the waiting time for data objects. The waiting time increases with the length of broadcast cycle. The second one is attributed to transaction restarts. Transaction restarts are mainly caused by data conflicts during a broadcast cycle. When the length of broadcast cycle increases, the number of transactions committed at the server per broadcast cycle increases. This in turn increases the likelihood of data conflict, which leads to a higher transaction restart rate.

4. Optimistic concurrency control in broadcast environments

In this section, we first describe the development of thought of our proposed protocol. Then, we briefly discuss the underlying principles, particularly regarding its validation. Then we describe the mechanism of the proposed protocol in a single broadcast disk environment.

4.1. Development of thought

In this study, we assume that the underlying concurrency control at the server is the conventional OCC. In order to develop a protocol that makes the least change to the server, one of the design objectives of the protocol is that the new protocol should be fully compatible to the OCC protocol running at the server. The key component in OCC protocols is the validation phase, where a transaction's destiny is decided. Validation can be carried out basically in two ways: *backward validation* and *forward validation* (Haerder, 1984). While in backward scheme, the validation process is done against committed transactions, in forward validation, validating of a transaction is carried out against currently running transactions. Forward validation provides flexibility for conflict resolution such that either the validating transaction or the conflicting active transactions may be chosen to restart, so it is more popular in database systems. In addition, forward scheme generally detects and resolves data conflicts earlier than backward validation, and hence it wastes less resources and time.

In wireless environments, there are different considerations. At the server, there are transactions submitted for validation by different sources including the mobile clients. It is not desirable to restart a validating mobile transaction because of high restart cost. Therefore,

forward validation is a better choice for the server because of the flexibility to choose those active transactions that are in conflict with the validating transaction to restart. In addition, only the write set of a transaction is required for forward validation, it allows the read-only transactions to be validated locally and autonomously at the mobile clients. As a result, the server takes the active role for the decision to commit transactions. In other words, the server determines the serialization order. On the other side, the mobile clients play a passive role. Given the serialization order determined by the server, the mobile clients have to determine whether their mobile transactions can be committed by detecting any data conflicts with the committed transactions. As a result, the mobile clients have to carry out the backward validation process. Although it is possible to carry out the backward validation at the end of a transaction execution, it causes delayed transaction restarts and wasted resources at the mobile clients. So, partial backward validation is introduced at the beginning of every broadcast cycle. Any transaction that is found to read inconsistent data is restarted immediately.

4.2. Principles

In our proposed protocol, transactions are allowed to execute unhindered until they reach the validation point or partial validation point for mobile transactions. The validation is based on the following principle to ensure the serializability.

If a transaction T_i is serialized before transaction T_j, the following two conditions must be satisfied:

Condition 1: *No overwriting*
The writes of T_i should not overwrite the writes of T_j.
Condition 2: *No read dependency*
The writes of T_i should not affect the read phase of T_j.

Generally, Condition 1 is ensured by performing the write phase serially in critical sections at the server. Thus, only Condition 2 will be considered. In the following sections, we will describe two validation schemes at the mobile clients and the server. The objective of these validation schemes is to satisfy the requirement of Condition 2.

4.3. Backward validation at the mobile clients

At the mobile clients, all transactions including read-only transactions and update transactions have to perform a partial validation at the beginning of every broadcast cycle. The partial validation is carried out by consulting the control information broadcast by the server at the beginning of every broadcast cycle. The content of the control information will be described later. If the transaction fails the partial validation, it will be aborted and restarted. Otherwise, the transaction can proceed. The partial validation process is carried out against committed transactions (at the server) in the last broadcast cycle. Data conflicts are detected by comparing the read set of the validating mobile transaction and the write set of committed transactions, since it is obvious that committed transactions precede the validating transaction in the serialization order. Such data conflicts are resolved to ensure Condition 2 by restarting the validating mobile transaction.

Let T_{pv} be the partial validating mobile transaction, and $CD(C_i)$ be the set of data objects that have been committed (updated) in the last broadcast cycle C_i at the server. Let $CRS(T_{pv})$ denote the current read set of transaction T_{pv}, which is the set of data objects that have been read by T_{pv} from previous broadcast cycles. The backward validation is described by the following procedure:

```
partial_validate(T_pv)
{
  if CD(C_i) ∩ CRS(T_pv) ≠ { } then
    abort(T_pv);
  else
  {
    record the value of C_i;
    T_pv is allowed to continue;
  }
  endif;
}
```

Note that $CD(C_i)$ is stored in the control information table. The value of C_i is recorded for the final validation to be described in the next section.

4.4. Forward validation at the server

At the server, validation of a transaction is done against currently running transactions. Note that a validating transaction at the server may be a server transaction or a mobile transaction submitted by the mobile clients. This process is based on the assumption that the validating transaction is ahead of every concurrently running transaction still in read phase in the serialization order. Thus, the write set of the validating transaction is used for data conflict detection to ensure Condition 2. The detection of data conflicts is carried out by comparing the write set of the validating transaction and the read set of active transactions. That is, if an active transaction, T_i, has read an object that has been concurrently written by the validating transaction, the value of the object used by T_i is not consistent. Such data conflicts are resolved by restarting the conflicting transactions in the read phase. For mobile transactions, data conflicts are detected in the partial validation described above.

For an update transaction submitted by a mobile client, it has to perform a final validation before the forward validation. The final validation is necessary because there may be transactions committed since the last partial validation performed at the mobile client. So, the broadcast cycle number of the last partial validation performed at the mobile client has to be sent to the server along with the update transaction for final validation.

Let T_v be the validating transaction, and T_a ($a = 1, 2, \ldots, n,\ a \neq v$) be the conflicting transactions existing at the server in their read phase. Let C_k be the broadcast cycle number received along with T_v. That is, T_v has performed it last partial backward validation at the mobile client in the broadcast cycle C_k. Note that a mobile transaction may read or write data objects after partial validation and before being sent to the server for final validation. Therefore, there may exist some transactions at the server that are committed and in conflict with T_v since its last partial validation. Let T_c ($c = 1, 2, \ldots, m$) be the transactions committed at the server since C_k. If the history of T_c is not available at the server due to space limitation, T_v has to be aborted. Let $\mathrm{RS}(T)$ and $\mathrm{WS}(T)$ denote the read set and the write set of a transaction T, respectively. Recall that $\mathrm{CD}(C_i)$ is the set of data objects that are updated in the current broadcast cycle and is initialized at the beginning of every broadcast cycle. Then the forward validation at the server is described by the following procedure:

```
validate(Tv)
{
  if Tv is a mobile transaction then
  {
    final_validate(Tv);
    if return fail then
    {
     abort(Tv);
     exit;
    }
  }
  foreach Ta (a = 1, 2, ..., n)
  {
    if RS(Ta) ∩ WS(Tv) ≠ { } then
     abort(Ta);
  }
  commit WS(Tv) to database;
  CD(Ci) = CD(Ci) ∪ WS(Tv);
}

final_validate(Tv)
{
  foreach Tc (c = 1, 2, ..., m)
  {
    if WS(Tc) ∩ RS(Tv) ≠ { } then
     return fail;
  }
  return success;
}
```

If T_v is successfully validated, the write set of T_v is recorded in the control information table, which will be broadcast in the next broadcast cycle. The mobile clients require the control information to perform the local partial validation. In addition, the final validation results (commit or abort) of the mobile transactions are included in the control information table for acknowledgement to the mobile clients for further action.

4.5. The roles of server and mobile clients

During each broadcast cycle, the server broadcasts the values of all data objects as well as the control information for the mobile clients to perform the partial validation. To ensure the serializability of all transactions submitted to the server including server transactions and update transactions submitted by the mobile clients, the server uses the OCC with forward validation protocol described above.

At the mobile clients, before any read or write operations are performed on data objects broadcast during a cycle, the control information transmitted at the beginning of the cycle is consulted to perform the partial validation. If the transaction fails the partial validation, the transaction is aborted. Otherwise, the read or write operation can proceed. A write operation on a data object is performed on a private workspace at the mobile client.

At the mobile clients, a read-only transaction can commit locally if it passes all the partial backward validation in the course of its execution. The transaction is serialized *after* all transactions committed before the beginning of the current broadcast cycle and is serialized *before* all transactions committed after the beginning of the current broadcast cycle. Note that a read-only transaction may be serialized before a transaction that is committed earlier at the server if the commit time of the transaction is later than the start time of the current broadcast cycle.

For an update transaction, the read set, write set, the updated values and the broadcast cycle number of the last partial validation performed at the mobile client have to be sent to the server for final validation because the position of the update transaction in the serialization order is determined on the fly (at the validation point) at the server. That is, the update transaction is serialized *after* all transactions committed before its validation point and is serialized *before* all active transactions. In the final validation, forward validation is required because those active transactions that have data conflicts with the validating transaction have to be identified and aborted. After the final validation, the result is sent back to the corresponding mobile client. To abort a transac-

tion at the mobile clients, all copies of the data objects written in the private workspace, if any, are discarded.

Theorem 1. *All committed transactions in the system are serializable.*

Proof. The correctness of this theorem follows directly from the arguments in the previous paragraph. □

Let us take the following set of transactions as an example with the schedule depicted in Fig. 1.

At the mobile client :	$Q_2 : r(a)r(b)r(c)$	$Q_3 : r(p)r(q)$	$U_4 : r(x)r(y)w(y)$
At the server :	$U_1 : r(a)w(a)$	$U_5 : r(q)w(q)$	$U_6 : r(y)w(y)$

In Fig. 1, we can find that Q_2 cannot pass the partial backward validation in broadcast cycle C_i. It is because after it has read data a and b from the broadcast cycle C_{i-1}, the value of a is written by U_1 before it can read data c in the broadcast cycle C_i and commits. On the other hand, Q_3 passes the partial backward validation and commits locally. Note that although Q_3 commits after U_5, Q_3 is before U_5 in the serialization order because Q_3 is not affected by U_5. For U_4 at the mobile client, it passes the partial backward validation in broadcast cycle C_i. After it reads the last data y and pre-writes the value of y in its private workspace, the required information for the final validation of U_4 is sent to the server. However, it fails the final validation because the data y is written by U_6 before U_4 reaches its validation point. If U_6 reaches its validation phase after U_4's validation point, U_6 will be aborted by U_4 in the forward validation process and U_4 can commit.

4.6. Contributions

In our proposed protocol, read-only transactions can be validated and committed locally at the mobile clients without contacting the server. In view of a large proportion of transactions being read-only transactions in most data dissemination applications, autonomy of the mobile clients to process read-only transactions saves much processing burden of the server and upstream communication cost. For update transactions, part of the validation process is performed in advance at the mobile clients and it alleviates part of the burden from the server.

In our proposed protocol, the partial backward validation helps to detect data inconsistency such that transactions can be aborted early at the mobile client side instead of at the end of read phase. In addition, the partial validation also helps to identify those update transactions that are likely to commit before transmitting them to the server for final validation such that the battery power can be used effectively. These are very desirable features in processing transactions to meet their deadlines in real-time applications.

The size of control information required in our proposed protocol is small. Let d be the size of a data object identifier, t be the maximum number of transactions committed in a broadcast cycle and N be the maximum number of (write) operations per transaction at the server. Note that the mobile clients are required to

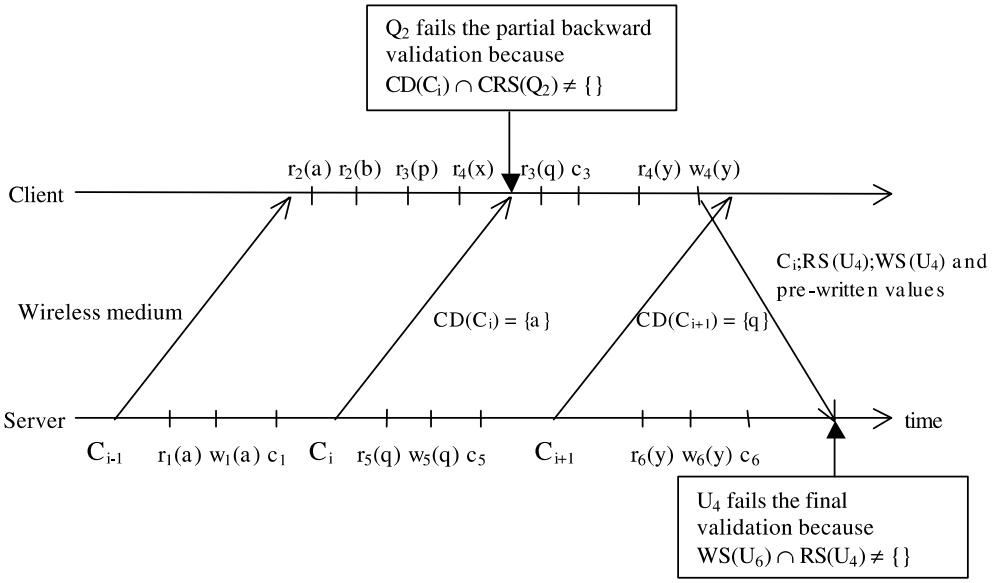

Fig. 1. Transaction execution schedule.

record the broadcast cycle number in which the partial validation of a mobile transaction is performed. So, the value of the broadcast cycle number has to be transmitted. Let c be the size of a broadcast cycle number. The total size of the control information is therefore $Ntd + c$.

5. Performance evaluation

The simulation experiments are aimed at studying the performance of our proposed protocol in contrast with the conventional OCC protocol in real-time broadcast disk environments. We did not consider the effects of caching in this performance study to rule out factors other than concurrency control. In other words, a mobile transaction may have to wait for the requested data object in the next broadcast cycle if the requested data object is missed (have been broadcast) in the current broadcast cycle.

The major performance metric of these protocols is the miss rate, which is the percentage of transactions missing their deadlines. Another performance metric is the restart rate, which is the average number of aborts and restarts before a transaction can be committed. Note that a transaction that has been restarted may meet its deadline. Therefore, this metric can demonstrate the effectiveness of the protocols in reducing transaction restarts. Transaction response time, which is the elapsed time when a transaction commits since a mobile client submits the transaction, is also collected. This metric includes the time involved in transaction restarts. Note that a transaction may be restarted more than once. The statistics of read-only mobile transactions (MROT) and update mobile transactions (MT) under conventional OCC and our proposed protocol (FBOCC) are collected separately.

5.1. Parameter setting

The simulation model is based on the model described in (Shanmugasundaram et al., 1999). It consists of a server, a client, and a broadcast disk for transmitting both the data objects and the required control information. At the server, conventional OCC protocol with forward validation is employed. The mobile clients process both read-only and update transactions. A deadline is assigned to every mobile transaction when they are submitted. The deadline of a mobile transaction is assigned as (*submission time* + *slack factor* × *predicted execution time*), where *predicted execution time* is a function of *transaction length*, *mean inter-operation delay* and *broadcast cycle length*. A mobile transaction is processed until it is committed, even the deadline is missed. The data objects that are accessed by a transaction are uniformly distributed in the database. Table 1 lists the baseline setting for the simulation experiments. These values are selected in order to create a scenario with high resource utilization and data contention. The number of read/write operations of a transaction is specified by the transaction length. The time unit is in bit-time, the time to transmit a single bit. For a broadcast bandwidth of 64 Kbps, 1 M bit-times is equivalent to approximately 15 s.

The server fills the broadcast disk with the data at the beginning of a cycle. Each cycle consists of a broadcast of all the data objects in the database along with the associated control information. The response times are measured in bit-time, and 95% confidence intervals were obtained with widths less than 5% of the point estimates of the response times.

5.2. Simulation results

Fig. 2 shows the miss rate of mobile transactions under different concurrency control mechanisms with

Table 1
Baseline setting

Parameter	Value
Mobile clients	
Transaction length (number of operations)	4
Read operation probability (for update transactions)	0.5
Fraction of read-only transaction	70%
Mean inter-operation delay	65,536 bit-times (exponentially distributed)
Mean inter-transaction delay	131,072 bit-times (exponentially distributed)
Slack factor	2.0–8.0 (uniformly distributed)
Server	
Transaction length	8
Transaction arrival rate	1 per 2,000,000 to 1 per 200,000 bit-times
Read operation probability	0.5
Number of data objects in database	300
Size of data objects	8000 bits
Concurrency control protocol	OCC with forward validation
Priority scheduling	Earliest deadline first

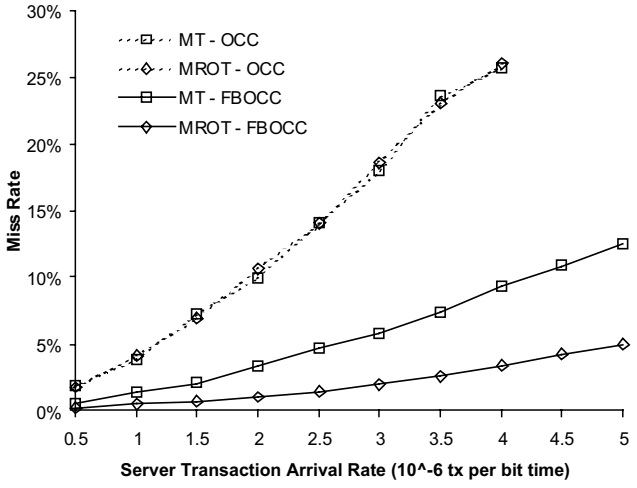

Fig. 2. Miss rate versus server transaction arrival rate.

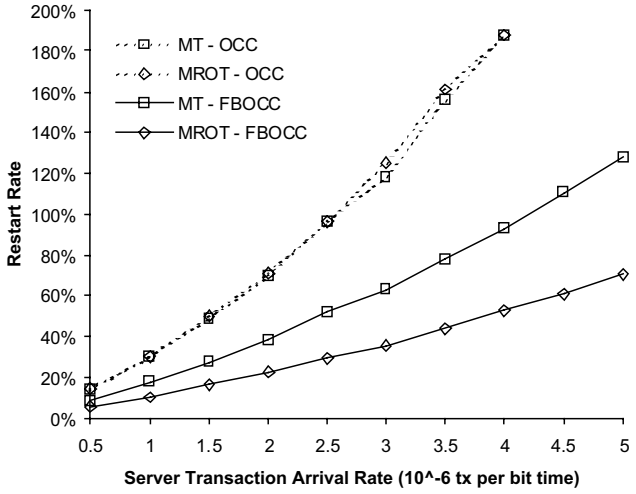

Fig. 3. Restart rate versus server transaction arrival rate.

different loadings at the server. For OCC, there is no discrimination between MROT and MT. Both are required to submit to the server for validation. Therefore, the performance of these two transaction types is almost the same. The miss rate increases with the server transaction arrival rate. Although the loading at the server is not high such that the impact of resource contention is not great, the increasing number of server transactions increases the chance of data conflicts between the mobile transactions and the server transactions. Consequently, mobile transactions submitted to the server for validation will be restarted if any data objects that have been read are over-written by a committed transaction. Since the delay overhead in broadcast environments is much higher than that in wired systems, it is more likely for a restarted transaction to miss its deadline.

On the other hand, the performance of mobile transactions under our proposed protocol (FBOCC) is significantly improved. The miss rate increases at a much slower rate with the server loading than OCC. In particular, the performance of MROT is even better than that of MT. The saving of validation of MROT at the server helps them to meet more deadlines. For MT, the partial validation at the mobile clients also helps to improve the performance.

As a restarted transaction may meet its deadline, the restart rate as shown in Fig. 3 and the response time as shown in Fig. 4 can help to give better understanding of the effectiveness of the protocols. Note that a transaction may be restarted more than once. For instance, mobile transactions are restarted more than once on average under OCC when the server transaction arrival rate is higher than 2.5. In fact, the transaction restart counts are almost the same for both protocols. However, the reduction in response time helps to increase the throughput under our proposed protocol such that the restart rate is effectively reduced.

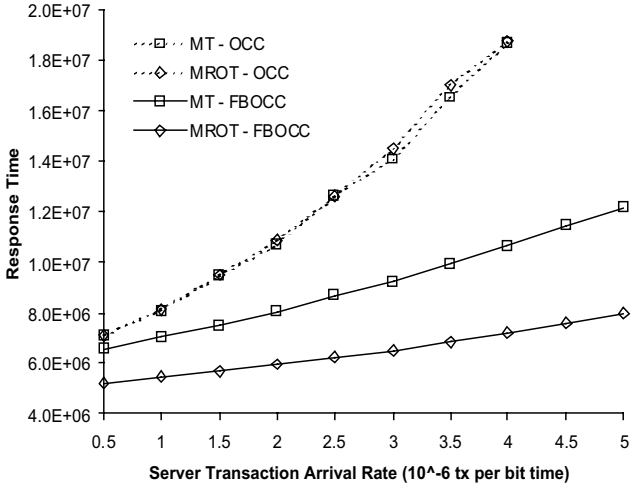

Fig. 4. Response time versus server transaction arrival rate.

6. Conclusions and future works

In this paper, we have proposed an efficient concurrency control protocol in real-time broadcast environments that is adapted from the conventional OCC protocols. The proposed protocol is developed under the requirements and constraints of real-time broadcast environments. The limited upstream bandwidth from the mobile clients to the server, the low bandwidth of wireless communication, the low capability, and the short battery life of mobile computers are considered in the design of the protocol. In this protocol, both read-only transactions and update transactions at the mobile clients are considered. At the mobile clients, read-only transactions can be processed and committed locally without contacting the server. For update mobile transactions, they are partially validated by consulting

the control information broadcast in every broadcast cycle. Only update transactions with the best chance to commit will be sent to the server for final validation. A series of simulation experiments have been conducted to study the performance of our proposed protocol compared to the conventional OCC protocol. The results showed that our protocol outperformed OCC for a wide range of parameter setting.

In conventional OCC protocols, forward validation is based on the assumption that the serialization order among transactions is determined by the arrival order of transactions at the validation phase. Thus, the validating transaction, if not restarted, always precedes concurrently running active transactions in the serialization order. This assumption is not necessary and can incur unnecessary transaction restarts. These restarts should be avoided (Lee and Son, 1993). In our future work, we will incorporate the mechanism of dynamic adjustment of serialization order in our new protocol in order to reduce transaction restarts. By using timestamp ordering mechanism, temporary serialization order among concurrently running transactions may be dynamically adjusted and recorded as far as data consistency is not violated.

Transactions submitted by the mobile clients may have priorities and data conflicts should be resolved in favor of higher priority transactions (Kuo et al., 1998; Lee et al., 1999) to enhance the real-time performance. Since forward validation provides flexibility for conflict resolution that either the validating transaction or the conflicting active transactions may be chosen to restart, so it is possible to introduce priorities in broadcast environments, as it may be more expensive to restart a mobile transaction than a server transaction. However, careful selection of transactions to abort is important to the overall real-time system performance because the cost to restart different transactions can be very different such as update transaction versus read-only transaction and mobile transaction versus server transaction.

In view of the nature of most data dissemination applications in real-time broadcast environments and the characteristics of wireless communications such as monetarily expensive bandwidth and slow transmission speed, transaction processing in broadcast environments is an excellent area for the application of data similarity (Chen and Mok, 1999b; Kuo and Mok, 1993). The similarity information gives both the mobile clients and the server the flexibility of adjusting the timing and frequency of data broadcast (Chen and Mok, 1999a). Cached values that satisfy certain similarity predicates can be used instead of accessing remote data source at the server. Similarity predicates can be some timing-related constraints that are used to check the validity of data values. Accordingly, the use of wireless communication and the number of transaction restarts can be reduced.

References

Acharya, S., Alonso, R., Franklin, M., Zdonik, S., 1995. Broadcast disks: data management for asymmetric communication environments. In: Proceedings of the ACM SIGMOD Conference, pp. 199–210.

Alonso, R., Korth, H., 1993. Database systems issues in nomadic computing. In: Proceedings of the ACM SIGMOD Conference, Washington, DC, pp. 388–392.

Barbara, D., 1997. Certification reports: supporting transactions in wireless systems. In: Proceedings of 17th International Conference on Distributed Computing Systems, USA, pp. 466–473.

Bernstein, P.A., Hadzilacos, V., Goodman, N., 1987. Concurrency Control and Recovery in Database Systems. Addison-Wesley, Reading, Massachusetts.

Bober, P.M., Carey, M.J., 1992. Multiversion query locking. In: Proceedings of the Eighteenth International Conference on Very Large Data Bases, Canada.

Bowen, T.F., Gopal, G., Herman, G., Hickey, T., Lee, K.C., Mansfield, W.H., Raitz, J., Weinrib, A., 1992. The datacycle architecture. Communications of the ACM 35 (12), 71–81.

Chen, D., Mok, A.K., 1999a. Building a similarity-based real-time database engine. In: Work in Progress of the 5th IEEE Real-Time Technology and Applications Symposium, pp. 35–38.

Chen, D., Mok, A.K., 1999b. SRDE—application of data similarity to process control. In: Proceeding of the IEEE 20th Real-Time Systems Symposium.

Dunham, M.H., Helal, A., Balakrishnan, S., 1997. A mobile transaction model that captures both the data and movement behavior. Mobile Networks and Applications 2, 149–162.

Garcia-Molina, H., Wiederhold, G., 1982. Read-only transactions in a distributed database. ACM Transactions on Database Systems 7 (2), 209–234.

Haerder, T., 1984. Observations on optimistic concurrency control schemes. Information Systems 9 (2).

Herman, G., Gopel, G., Lee, K.C., Weinrib, A., 1987. The datacycle architecture for very high throughput database systems. In: Proceedings of the ACM SIGMOD Conference, pp. 97–103.

Huang, Y., Sistla, P., Wolfson, O., 1994. Data replication for mobile computers. In: Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, pp. 13–24.

Imielinski, T., Badrinath, B.R., 1994. Mobile wireless computing: challenges in data management. Communications of the ACM 37 (10), 18–28.

Kung, H.T., Robinson, J.T., 1981. On optimistic methods for concurrency control. ACM Transactions on Database Systems 6 (2), 213–226.

Kuo, T.W., Kao, Y.T., Shu, L., 1998. A two-version approach for real-time concurrency control and recovery. In: Proceeding of the Third IEEE International High Assurance Systems Engineering Symposium, Washington, DC, November.

Kuo, T.W., Mok, A.K., 1993. SSP: a semantics-based protocol for real-time data access. In: Proceedings of IEEE Real-Time Systems Symposium.

Lam, K.W., Son, S.H., Lee, V.C.S., Hung, S.L., 1998. Using separate algorithms to process read-only transactions in real-time systems. In: Proceedings of IEEE Real-Time Systems Symposium.

Lam, K.Y., Au, M.W., Chan, E., 1999. Broadcast of consistent data to read-only transactions from mobile clients. In: Proceedings of Second IEEE Workshop on Mobile Computing Systems and Applications.

Lee, J., Son, S.H., 1993. Using dynamic adjustment of serialization order for real-time database systems. In: Proceedings of 14th IEEE Real-Time Systems Symposium, pp. 66–75.

Lee, V.C.S., Lam, K.Y., Kao, B., 1999. Priority scheduling of transactions in distributed real-time databases. Real-time Systems 16 (1), 31–62.

Lee, V.C.S., Lam, K.Y., Tsang, W.H., 1998. Transaction processing in wireless distributed real-time database systems. In: Proceedings of the 10th Euromicro Workshop on Real Time Systems, Berlin, June, pp. 214–220.

Peng, C.S., Lin, K.J., 1996. A semantic-based concurrency control protocol for real-time transactions. In: Proceedings of the Second IEEE Real-Time Technology and Applications Symposium, pp. 59–67.

Pitoura, E., 1998. Supporting read-only transactions in wireless broadcasting. In: Proceedings of the DEXA98 International Workshop on Mobility in Databases and Distributed Systems, pp. 428–433.

Pitoura, E., Bhargava, B., 1994. Building information systems for mobile environments. In: Proceedings of the Third International Conference on Information and Knowledge Management, pp. 371–378.

Pitoura, E., Chrysanthis, P.K., 1999. Scalable processing of read-only transactions in broadcast push. In: Proceedings of the 19th IEEE International Conference on Distributed Computing System.

Pitoura, E., Samaras, G., 1998. Data Management for Mobile Computing. Kluwer Academic Publishers.

Shanmugasundaram, J., Nithrakasyap, A., Padhye, J., Sivasankaran, R., Xiong, M., Ramamritham, K., 1997. Transaction processing in broadcast disk environments. In: Jajodia, S., Kerschberg, L. (Eds.), Advanced Transaction Models and Architectures. Kluwer, Boston, pp. 321–338.

Shanmugasundaram, J., Nithrakashyap, A., Sivasankaran, R., Ramamritham, K., 1999. Efficient concurrency control for broadcast environments. In: ACM SIGMOD International Conference on Management of Data.

Stankovic, J.A., Son, S.H., Hansson, J., 1999. Misconceptions about real-time databases. Computer 32 (6), 29–37.

Stankovic, J.A., Zhao, W., 1988. On real-time transactions. SIGMOD Record 17 (1), 4–18.

Yu, P.S., Wu, K.L., Lin, K.J., Son, S.H., 1994. On real-time databases: concurrency control and scheduling. Proceedings of the IEEE 82 (1), 140–157.

Zdonik, S., Alonso, R., Franklin, M., Acharya, S., 1994. Are disks in the air just pie in the sky. In: Proceedings of the Workshop of Mobile Computing Systems and Applications, California.

Victor C.S. Lee received the Ph.D. degree in Computer Science from the City University of Hong Kong in 1997. He is now an Assistant Professor in the Department of Computer Science of the City University of Hong Kong. He has been a Visiting Research Associate in the Department of Computer Science of University of Virginia. His research interests include transaction processing, mobile computing, and real-time databases. He is a member of the IEEE and the IEEE Computer Society.

Kwok-Wa Lam received the Ph.D. degree in Computer Science from the City University of Hong Kong in 1997. He received the B.Sc. degree with first class honors in 1993 from the same university. He is now a Research Fellow in the Department of Computer Science of the City University of Hong Kong. His research interests include real-time databases, mobile computing, and data mining.

Tei-Wei Kuo received B.S.E. degree in computer science and information engineering from National Taiwan University in Taipei, Taiwan, in 1986. He received the M.S. and Ph.D. degrees in computer sciences from the University of Texas at Austin in 1990 and 1994, respectively. He is currently a Professor and the Vice Chairman in the Department of Computer Science and Information Engineering of the National Taiwan University, Taiwan, ROC. He was an Associate Professor in the Department of Computer Science and Information Engineering of the National Chung Cheng University, Taiwan, ROC, from August 1994 to July 2000. The research interests of Professor Kuo include real-time databases, real-time process scheduling, real-time operating systems, and embedded systems. He is the Program Co-Chair of IEEE 7th Real-Time Technology and Applications Symposium, 2001, and an associate editor of the Journal of Real-Time Systems since 1998. He has consulted for government and industry on problems in various real-time systems design. Dr. Kuo is a senior member of the IEEE computer society.