

A Multi-Version Data Model for Executing Real-time Transactions in a Mobile Environment

Kam-Yiu Lam¹, GuoHui Li¹ and Tei-Wei Kuo²

Department of Computer Science¹
City University of Hong Kong
83 Tat Chee Avenue, Kowloon
Hong Kong

Department of Computer Science and
Information Engineering²
National Taiwan University
Taipei, Taiwan, ROC

ABSTRACT

With the significant advances in mobile computing technology, there is an increasing demand for various mobile applications to process transactions in a real-time fashion. When remote data access is considered in a mobile environment, data access delay becomes one of the most serious problems in meeting transaction deadlines. In this paper, we propose a multi-version data model and adopt the relative consistency as the correctness criterion for processing of real-time transactions in a mobile environment. The purpose is to reduce the impacts of unpredictable and unreliable mobile network on processing of the real-time transactions. Under the proposed model, the overheads for concurrency control can be significantly reduced, and the data availability is much enhanced even under network failures. A real-time transaction may access stale data, provided that they are relatively consistent with the data accessed by the transaction, and the staleness of the data is within the requirements. An image transaction model, which pre-fetches multiple data versions at fixed hosts, is proposed to reduce the data access delay and to simplify the management of the real-time transactions in a mobile environment. The image transaction model also helps in reducing the transaction-restart overheads and minimizing the impacts of the unpredictable performance of mobile network on transaction executions.

1. INTRODUCTION

In recent years, there has been growing interest in the design of efficient transaction processing methods for *mobile real-time database systems (MRTDBS)* [KU99, LKT00, U98]. A MRTDBS is, in general, defined as a *real-time database system (RTDBS)* over a mobile network, where a RTDBS is defined as a database system in which transactions have timing constraints, such as deadlines on their completion times [R93, YWLS94]. Any failure in meeting the timing constraints may seriously reduce the usefulness of transactions' results. For example, a late response to a stock query may result in the loss of a good trading opportunity.

A MRTDBS may consist of a collection of mobile hosts and fixed hosts. The database is, then possibly, partitioned and managed by each individual mobile host as well as the fixed hosts. The mobile hosts communicate with the fixed hosts over a mobile network, and the fixed hosts might be connected together by a high-speed wired network. In many cases, there are good reasons to maintain a database at a mobile host. For example, a mobile host may be associated with sensors and/or data-acquisition devices so that they can record the status of selected objects in the external environment in a real-time fashion, while the mobile host moves around. Furthermore, the pre-processing of data at mobile hosts can significantly reduce the server workloads and provide quick and effective services to users in many ways [BGS01]. In such a distributed environment, mobile hosts may initiate real-time transactions (in fact, by users), called *mobile real-time transactions*, to read and update data items residing at fixed hosts and/or mobile hosts. Example applications of MRTDBS are tele-medicine systems, real-time traffic information and navigation systems, and mobile Internet stock trading services, etc. For example, in an air navigation system, the airplane may maintain a real-time database to record the current status of the airplane, e.g., its coordinate and fuel level, and the weather condition. The pilot may initiate a transaction to find out its current coordinate and the coordinates of other airplanes and airports within certain diameter. A local (mobile) database on the airplane can be used to record the trace of the flight and important information all the way to the destination.

Owing to the intrinsic limitations of a MRTDBS, the design of an efficient and cost-effective MRTDBS requires techniques that are very different from those developed for a *distributed real-time database systems (DRTDBS)*, which is supported on a wired network [LKT00]. Compared with wired networks, mobile networks are usually much slower and unreliable [LC00]. Disconnection between a mobile host and a fixed host is very common [IB94]. It is much more difficult to meet transaction deadlines in a mobile environment. The communication delay for the processes of a mobile transaction is unpredictable and usually pretty long, and the cost in resolving data conflicts among real-time transactions becomes much higher. For example, if a restart-based method is adopted in a MRTDBS, e.g., the High Priority Two Phase Locking [AGM92], the restart of a transaction could take a long time since the processing of the restarted transaction may involve fixed hosts as well as mobile hosts. The temporal constraints of data items in a real-time database are difficult to maintain [R93]. Furthermore, the mobility of mobile hosts can affect the workload distribution in the system, and it certainly complicates the design of transaction scheduling algorithms, when transaction deadlines/response times are considered. Another

serious problem in a MRTDBS is the potential risk of network disconnection, which can seriously affect the management of a transaction, especially when it has different sub-transactions executing across a disconnected network. Not only the processing of the disconnected transaction will be affected, other transactions may also be affected if they want to access the data items currently accessed by the disconnected transaction.

In the past few years, various techniques have been proposed for processing transactions in a mobile environment. Most of the proposed methods suggested to relax the consistency requirements for transactions and to break down a mobile transaction into smaller sub-transactions to resolve the network disconnection problem. In [PB95], the transactions are classified into strong and weak transactions, and the system is divided into clusters. It is suggested that a weak transaction may commit even if it observes inconsistent data values, provided that the degree of inconsistency is within the user acceptable limit. Another similar technique is to divide a mobile transaction into sub-transactions such that data consistency and atomicity are merely applied to each sub-transaction, instead of the entire transaction. Although these techniques are useful for conventional mobile database systems, they may not be suitable to MRTDBS that are mostly mission-critical applications. For some cases, even a small degree of inconsistent may generate very harmful effects. In other words, the user acceptable error margin is very small. Consider an air navigation system. If the current coordinates of an airplane and another approaching airplane in the system reflect their coordinates at different time points, the pilot may make a wrong flight decision. Furthermore, many (real-time) transactions cannot be divided into independent sub-units and/or sub-transactions.

In this paper, our objective is to explore the technology in executing *mobile real-time transactions (MRTT)* efficiently, where MRTT's are initiated at mobile hosts in a MRTDBS. The critical issue is how to minimize the impacts of a mobile network on the probability of deadline violation and, at the same time, to provide *consistent* data items to transactions. Instead of breaking a MRTT into sub-transactions and providing inconsistent data items to a transaction (in case of a network error), we propose another alternative to resolve the problem by providing *relatively* consistent data values to a transaction, i.e., reading older versions of data items which are relatively consistent. Note that the relative consistency requirement defined in this paper can work together with other relaxation methods such as epsilon serializability and similarity [KM92]. We hope that by introducing one more dimension for relaxing the transaction consistency requirement, the user requirements and transaction deadline requirements can be better satisfied. We must emphasize that, in many real-time applications, reading of stale data items is acceptable, provided that they are within the user staleness requirement. We are interested in soft real-time transactions in this study, and their characteristics are predictable with a high-degree of certainty, where a soft real-time transaction is a real-time transaction with a soft deadline.

2. RELATED WORK

The research in RTDBS has received a lot of attention in the past decade. Researchers have proposed various techniques in the design of a database system to improve and to guarantee the system performance. The majority of the work is in the design of efficient concurrency control protocols to meet transaction

deadlines and, at the same time, to maintain database consistency [AGM92, HLC90, OS95, SPL92]. Researchers often assumed that the behavior and/or characteristics of transactions in RTDBS are more predictable than those in conventional database systems [KS96]. Different transaction scheduling algorithms, earliest deadline first (EDF), are designed to meet the specific requirements of different types of transactions.

Another important characteristic of RTDBS's is on the temporal properties of certain data items, where their values change rapidly with time. These data items are usually used to record the status of the corresponding objects in the external environment. Various concepts, such as external consistency and relative consistency, have been proposed to define the correctness of temporal data items, e.g. [KLA99, R93]. External consistency requires the values of data items to reflect the current status of the corresponding objects in the external environment. In order to maintain external consistency, updates (or transactions) must be generated to refresh the values of the data items if the corresponding objects change their status. If the difference between a data item and the corresponding object's status exceeds a pre-defined tolerance limit, the system becomes *external inconsistent*. External consistency can be realized by defining an absolute validity interval (avi) for each temporal data item under consideration. The avi of a data item is usually defined based on user requirements and/or by assuming the maximum rate of changes in data values. Relative consistency defines the consistency for data items that are accessed by the same transaction. In other words, when a group of data items are accessed by a transaction, their values should reflect the status of their corresponding objects approximately at the same time. In [KLA99], absolute and relative consistency is defined for temporal data items with discrete values. Under the proposed temporal consistency requirements, the data items accessed by a transaction have to be valid at the same time point.

With the advances in mobile computing technology, conventional information services face serious challenges in how to deliver existing and even more innovative services over mobile network. Such market needs triggers the research in mobile RTDBS. In [LKT00], the transaction shipping approach is proposed to reduce the impact of mobile network on the system performance for MRTDBS, where the databases are maintained over wired and wireless networks. By the pre-analysis, the impacts of the dynamic properties of transaction on the system performance can be minimized. The concept of data similarity [KM92] is proposed for concurrency control in MRTDBS in [LKL99], and a similarity-based High Priority Two Phase Locking protocol is proposed. It has been found that the performance of the system can be significantly improved under the proposed protocol, and the improvement depends on the return probability of the transactions, i.e., the number of communication runs between the originating mobile client and the fixed host, and the read operation probability. A different transaction processing architecture, called Execution at Fixed Host (EFM) and Execution at Mobile Host (EMH), are proposed and examined for MRTDBS [U95]. It has been found that EFM could give a better performance, compared to EMH due to lower communication overheads.

3. SYSTEM MODELS

A MRTDBS consists of mobile hosts (MH's), fixed hosts (FH's), mobile supporting stations (MSS's), a fixed network, and a mobile network, as shown in Figure 1. FH's are connected with each other by a reliable high-speed fixed network. The geographical area of the system is divided into a number of interconnected cells, and there is a MSS in each cell to communicate with the MH's in the cell. A MH communicates with a fixed FH through the MSS of its current cell.

The databases are partitioned and managed by the MH's and the FH's. It is assumed that for each data item maintained by a MH, a backup copy is also maintained by a FH. After an update is performed at the primary copy of a data item residing at a MH, the corresponding backup copy at the FH will be updated accordingly. Note that the synchronization of the values between the primary copy and the backup copy of a data may be affected, due to network disconnection. The purposes of duplicating a data item at a fixed host are to improve the data availability and the reliability of the system. The primary copy of a data item maintained by a MH cannot be reached if the MH is disconnected from the network. With the backup copy, a transaction may continue its execution by accessing the backup copy at the FH.

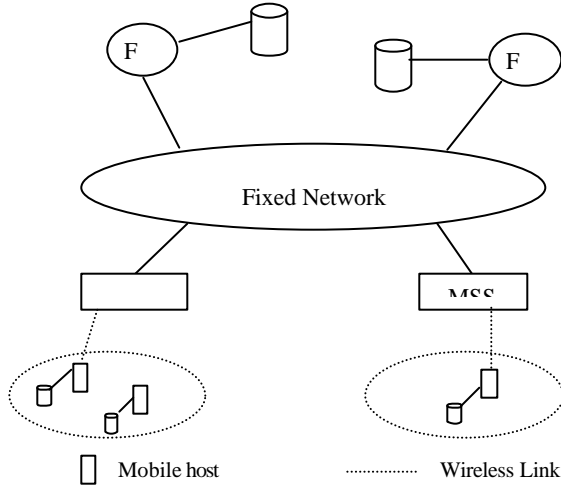


Figure 1: Architecture of a MRTDBS

The database consists of two types of data items: *temporal* and *non-temporal*. The validity of a temporal data item changes with time, and their values are usually high dynamic. A typical example of a temporal data item is the location of a moving object. The values of non-temporal data items are less dynamic, and they are usually used to record static information which changes infrequently, e.g., the location and the name of an airport. Since the maintenance of non-temporal data items is not the main concern in this paper, we assume that all the data items considered in this paper are temporal data items.

Based on the type of information that a data item represents, temporal data items are classified into two types: *base items* and *derived items* [KLA99]. Base items are used to record the status of the corresponding objects in the external environment, e.g., the last traded price of a stock and the location of a moving object. We are interested in applications, where new value updates of base items are generated by mobile hosts (for those generated by

fixed hosts, techniques developed in this paper can be further extended). To maintain consistency between the value of a base item and the status of the corresponding object in the external environment (called external consistency in some literature), updates (or transactions), which capture the new status of the object, have to be initiated by mobile hosts to refresh the value of the base item.

The value of a derived item is derived from base items and/or other derived items. A typical example of a derived item is the safety level of a flying flight. In this paper, we assume that the value of a derived item is derived from a pre-defined set of base items and/or derived items, and is created by MRTTs. Note that the functions of a MRTT are usually much more pre-defined than the transactions in a conventional database system [KS96]. It is usually assumed that MRTT's are simple and flat transactions, and each MRTT consists of a sequence of read operations on base items, and the last operation is a write operation on a derived item. In between the operations of a MRTT, control statements are defined to control the logic flow of the transaction. It is assumed that the functions and behavior of the set of MRTT's from the mobile hosts are pre-defined and maintained at the fixed hosts. The MRTT's in the system are classified into different types based on their functions. Therefore, once the transaction type is identified, the set of operations, logical structure of the transaction can be defined at the fixed host. Each MRTT is given a deadline, and it is assumed that the EDF algorithm is used for CPU scheduling. Formally, a MRTT T_i is defined by a 5-tuple:

$$T_i = (C, O_i, D_i, <_i, \Delta_i),$$

where C_i = the transaction type of T_i
 O_i = the set of operations of T_i
 D_i = the transaction deadline of T_i
 $<_i$ = the partial order relationship defined among the operations of T_i
 Δ_i = the maximum degree of data staleness that T_i can tolerate. It is assumed that a MRTT may accept a data item which is not the most current one, provided that it is not "too old" and *relative consistent*. The staleness of a data item is defined as the current time minus the latest valid time of the data item. If the difference is smaller than Δ , then it is within the staleness tolerance limit of the transaction. (For the detailed discussion and definition, please refer to Section 4.2.)

A MRTT may need to access data items located at fixed hosts as well as mobile hosts. It is the scope of this paper to study an efficient method to process MRTTs such that the deadline constraints and consistency requirements of the transactions can be satisfied, even over an unreliable mobile network environment. A multi-version data model will be introduced in the next section. The execution model for the transaction will be discussed in Section 5.

4. DATA MODEL AND TEMPORAL CONSISTENCY

In this paper, we propose a multi-version data model to support the processing of MRTTs, i.e., the FH not only manages the backup copy of the current version, but it also maintains the previous backup copies of the data item. Note that due to the synchronization problem between the mobile host and the fixed

host, some of the backup copies may still maintained at their originating mobile host. The transmission of backup copies from a mobile host to a fixed host can only be done after the re-establishment of a mobile channel between them. Once a copy has been transmitted to a FH, the MH may remove it if it is not the latest version.

With the multi-version data model, we define relative consistency as the temporal correctness criteria for the execution of a MRTT. Maintaining multi-versions for a data item can increase data availability. If a mobile host is temporarily disconnected, the execution of a MRTT will be affected if it wants access to a data item maintained by the mobile host. In order to save a MRTT from missing its deadline, it may access an old version of the data item at a FH.

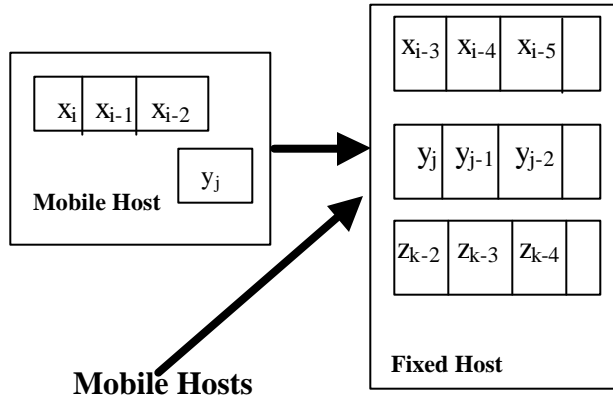


Figure 2: Multi-version Data

4.1 Multi-version Data Model

Under the multi-version data model, each data item consists of a sequence of versions, i.e., x_i is the i^{th} version of data item x . A validity interval (VI) is defined on each version. For example, the validity interval of x_i is denoted as $VI(x_i) = [LTB(x_i), UTB(x_i)]$, where LTB and UTB are the *lower time bound* and the *upper time bound* of x_i respectively. The validity interval of a version indicates at which time interval the version is valid.

For a base item, new versions are created by updates at a mobile host. When an update is generated, it is associated with a time-stamp, which indicates at which snapshot its value is representing. When the update is completed, the LTB of the new version of the item will be set as the time-stamp of the update and the UTB of new version will be set to infinite temporarily until the next update arrives. At the same time, the UTB of the latest version of the data item before the update will be set as the time-stamp of the update. Thus, the validity intervals of two versions of a base item does not overlap with each other and the validity intervals of successive versions continue the validity of the last versions, i.e., the UTB of a version is the LTB of the next version.

The new version of a derived item is created by a MRTT, which reads base items and then derives a new value for the derived item. To be temporally consistent, the validity interval of a derived item is determined from the validity intervals of the set of base items from which its value is derived from, i.e.,

Definition: The validity interval (VI) of a derived item dx_i is defined from MRTT T , which update the value of dx_i . VI of dx_i is

defined as the intersection of the validity intervals of all the base items read by T to derive the value for dx_i , i.e.,

$$VI(dx_i) = \bigcap \{VI(x_j) \mid x_j \in DS(T)\},$$

where $DS(T)$ is the read set of T for calculating dx_i .

If the UTB of a base item is undefined, i.e., equals to infinity, the time when the base item is accessed by a MRTT will be used by the MRTT for the calculation of the VI of a derived item. Although using the access time is a pessimistic approach and the actual UTB of the item should be greater, this can eliminate the requirement to update the VI of the derived item later when the UTB of any of its base items are defined.

4.2 Temporal Data Consistency

Temporal consistency refers to how well the value of a data item models the actual status of the external environment. This is critical to most real-time database applications since many of them are reactive systems, i.e., generate timely responses to the changes in the external environment. Basically, temporal consistency consists of two components: *absolute consistency* and *relative consistency*. A transaction observes absolute consistency if all its accessed data items truly reflect the status of the corresponding objects in the external environment. Absolute consistency constraints are usually defined on base items. The determination of absolute consistency of a base item can be based on unapplied update, i.e., the value of a data item is stale when a new update for the data item is generated. A set of data items is relatively consistent if they are temporally correlated to each other, i.e., representing the status at the same time point. Relative consistency constraints are usually defined over a set of base items with a derived item which value is derived from the set of base items.

As discussed in many previous literatures in RTDBS, it is not easy to maintain temporal consistency in a RTDBS, especially absolute consistency. The problem will be more difficult to be resolved in a mobile environment. For example, after a MRTT T has accessed the latest version of a temporal data item and then sends a request to a mobile host to access the latest version of another data item. Because of network disconnection or network delay, the data version can only be transmitted to the requesting transaction after a long delay. During this period of time, a new version of the previously accessed data item may be created. To maintain absolute consistency following the unapplied update requirement, the transaction has to be restarted. It is obvious to see that if the rates of changes in values of the temporal data items are high, a transaction may be repeatedly restarted for many times especially if they are long transactions.

In order to reduce the number of transaction restarts and increase the probability of committing a MRTT before its deadline, we may relax the absolute consistency requirement. A transaction may consider its accessed data item valid if the data item is absolutely consistent at the time when it accesses the data item instead of requiring it to be valid until its commitment. However, this relaxation may product inconsistent results. For example as shown in Figure 3, a MRTT T reads a data item x_i at time t_1 and x_i becomes invalid at time t_2 . Later, T reads y_2 at time t_3 . It is obvious to see that T may observe inconsistent data values since the version y_2 may be generated by a transaction which produces x_2 .

To resolve the problem, we need to enforce the relative consistency requirement such that all the data items accessed by a transaction are absolute consistent at the same time point, i.e., either x_j and y_j or x_2 and y_2 . At the time to satisfy the staleness requirement of a transaction, the time point should not be older than $(st - \Delta)$, where st and Δ are the start time and staleness requirement of the transaction. Otherwise, they are too old to be useful. Note that to most temporal data, their usefulness will decrease as the time passes by as new versions are created. In the following, we define the relative consistency.

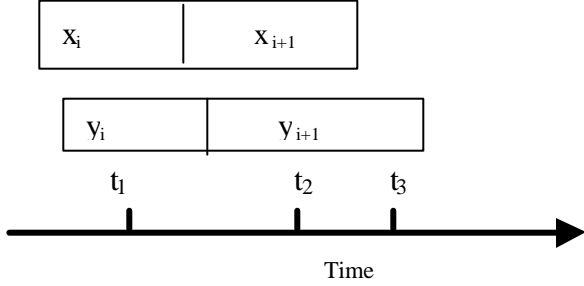


Figure 3: Inconsistent data items

Definition of relative consistency: A transaction with a maximum staleness Δ , start time st , commit time ct and a read data set R observes relative temporal consistency if $\exists t, (st - \Delta) \leq t \leq ct, \forall O \in R, LTB(O) \leq t \leq UTB(O)$.

The above relative consistency requirement will be adopted in the design of transaction model and execution strategies for the transactions, and will be discussed in the next section.

5. PROCESSING OF MOBILE REAL-TIME TRANSACTION

In this section, we propose an image transaction model to process MRTTs under the multi-version data model. The purpose is to reduce the data access delay especially those due to network errors, and at the same time to provide *relatively* consistent data items to the transaction. Two important advantages of the image transaction model are:

- (1) since the MRTT is processed at a fixed host, called anchor host, and all its required data items are also located at the fixed host, the overhead for transaction restart due to inconsistent data values is small; and
- (2) the overhead for the management of the transaction will be low and less affected by the unreliable network and network disconnection.

5.1 Image Transaction Model Method

In the image transaction model, a transaction, called *image transaction*, is generated once a MRTT is received by a fixed host which is called the *anchor host* of the MRTT. The main job of the image transaction is to represent the MRTT to get its required data items to the anchor host such that every time, when the operations of the MRTT want to access a data item, they can find the data item at the anchor host. Of course, the success of the definition of the image transaction lays on how to predict the operations and the required data items of the operations of the MRTT. This can be achieved by performing a pre-analysis once the fixed host receives the MRTT. Fortunately, for most MRTDBS, the set of

MRTTs to be generated from the HMs and the characteristics of the transactions are not difficult to predict due to their pre-defined functions and properties.

The execution model of a MRTT is as shown in Figure 4 and its execution can be divided into 3 steps:

Step 1 – Generation of Image Transaction: Once the anchor host receives the MRTT, a pre-analysis procedure will be performed to generate the image transaction which consists of a set of image sub-transactions. The sub-transactions of the image transaction will be sent to other fixed hosts where the required data items of the MRTT are resided.

Step 2 – Data Pre-fetching by image sub-transaction: Sub-transactions of the image transaction are performed concurrently at fixed hosts to pre-fetch data items to the anchor host. In the execution of an image sub-transaction, its first checks with the mobile hosts to determine whether any new versions of its required data items have been created at the mobile hosts. If there are more updated versions of a data item have been generated, it will ask the mobile host to send the new versions to the fixed host. If it cannot contact the mobile host, it will try again later until its deadline is expired. In case, a new version has been created by the mobile host after it has transmitted the versions to the anchor host, it will forward the new version to the anchor host once it has received. After the transaction is completed, the results will be sent to the originating MH upon its commitment. So, we can see that responsibility of getting data for the execution of a MRTT is downloaded to the image transaction.

Step 3 – Execution of the MRTT: After the transmission of image sub-transactions to fixed hosts, the execution of the MRTT may start. While it is executing, its required data items will be transmitted by the image sub-transaction to the anchor host. Relative consistency of the data items will be checked while it is executing.

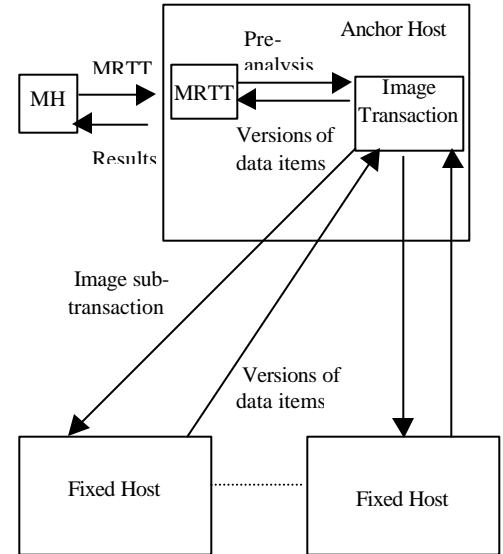


Figure 4: Structure of the Image Transaction Model

5.1.1 Generation of Image Transaction

The generation of an image transaction is based on the pre-analysis procedure performed upon the received of an MRTT. The pre-analysis is performed at the anchor host, which is also responsible for the execution of the MRTT. The pre-analysis procedure first reads the transaction type of the received MRTT and then based on the pre-defined transaction profile to determine the operation set of the transaction. The deadline and staleness requirement of the image transaction comes from the originating MRTT.

After the pre-analysis, an image transaction t is defined for a MRTT T . It is defined as a 3-tuple:

$$\text{ImagTran}(t) = (P_t, D_t, \Delta_t)$$

where P_t = the set operations in t

D_t = the deadline of t .

Δ_t = the staleness requirement of t .

Note that since the purpose of an image transaction is to retrieve data items for the MRTT, all the operations in an image transaction are read operations and they are a sub-set of the operations of its originating MRTT. The staleness requirement and deadline of an image transaction are the deadline and staleness requirement of its originating MRTT, respectively. Δ_t is used by the image transaction to determine which versions of data items are pre-fetched from the fixed hosts for the execution of the MRTT. A version will be useless if it is too old.

In the system, it is assumed that earliest deadline first (EDF) is used for assigning priorities to MRTTs. To favor the data pre-fetching process, the priorities of all the image transactions are assigned to be higher priorities than all the MRTTs. For the scheduling amongst image transactions, the EDF is adopted and the image transaction from a MRTT with a closer deadline will be assigned to a higher priority.

After defining the set of data items for an image transaction, the sub-transactions of the image transaction will be defined based on the data set of the image transaction and the distribution of the data items in the system. The data items in an image transaction are partitioned into several sub-sets, P_1, P_2, \dots, P_n , such that all the data items in P_i resides at the same fixed host FH_i . Each image sub-transaction is a process at a specific fixed host to pre-fetch the versions of the data items of its data set to the anchor host of its originating image transaction. The deadline and staleness requirement of an image sub-transaction is set to be that of its originating image transaction. Note that the set of image sub-transactions can be performed in parallel at different fixed hosts.

5.1.2 Version Selection

Since each data item has multiple versions maintained at a fixed host, pre-fetching all the versions of a data item will represent a heavy workload and at the same time not all the versions are suitable. They need to satisfy two constraints: (1) the staleness requirement and (2) the relative consistency requirement, of the MRTT. The first requirement will be checked in the version selection procedure while the second requirement is checked while a MRTT is executing.

The version selection procedure consists of two parts. Firstly, once the image sub-transaction P_i arrives a fixed host, FH_i , it will send out messages to the mobile hosts which are responsible for generating updates for its required data items to ask them to

forward any new versions of its required data items. While it is waiting for the reply from the mobile hosts, it gathers appropriate versions for its data items at fixed host FH_i . For the originating MRTT T with start time st , to satisfy the relative consistency requirement, the transaction should not access data items which are invalid before $(st - \Delta_t)$. Therefore, it is unnecessary to pre-fetch the data versions which *upper time bounds* are less than $(st - \Delta_t)$. After it has received all the replies and new versions from the mobile hosts or after the waiting time for reply is expired, it will determine the UTB of the latest versions of its required data items as followings:

- (1) For those data items, which mobile hosts have replied to the image sub-transaction, the UTB of the latest version will be set as the current time.
- (2) For those data items, which mobile hosts did not reply to the image sub-transaction, the UTB of the latest version will be set as *undefined* and the version will be discarded.

After assigning the UTB of the latest versions of its required data items, the set of data versions will be sent to the anchor host of its originating MRTT. Then, the image sub-transaction will be destroyed if all its mobile hosts have already replied to it. Otherwise, it will stay to ask any pending mobile hosts until its deadline is expired.

5.1.3 Execution of Mobile Real-time Transaction

The execution of a MRTT may start once the required data items of its first operation are available at its anchor host. Since its image transaction will provide its required data items, including different versions, for its execution, it does not need to submit additional data requests while it is executing. In order to meet the relative consistency requirement of a transaction, all the versions of the data items accessed by the transaction have to be relatively consistent at the same time point and the time point must not smaller than $(st - \Delta)$ where st and Δ are the start time and the maximum staleness of the transaction, respectively. To ensure the above conditions will be satisfied, before the execution of a MRTT, T , it is defined with a transaction interval based on its st and Δ such that its LTB_T , LTB_T , equals to $(st - \Delta)$ and its UTB_T , equals to infinity.

In executing the operations of a MRTT, the latest version of its required data items from the set of data items collected from its image transaction will be chosen. After MRTT T accesses a version, x , its transaction interval will be adjusted as following:

$$LTB_T = \max(LTB_T, LTB_x) \text{ and} \\ UTB_T = \min(UTB_T, UTB_x)$$

When it selects the version for its next data items, the following conditions have to be met:

- (1) the UTB of the version must be greater than its LTB_T ; and
- (2) the LTB_T of the version must be smaller than its UTB_T .

If there does not exist such a version, the transaction will be restarted and the next later version of its first accessed data item will be selected. Then, the above procedure will be repeated until all its operations have been completed. Note that even though a transaction has to be restarted, the overhead will not be very heavy since all its required versions are already exist in the anchor host.

5.2 Stepwise Data Pre-fetching

It is obvious to see that the effectiveness of the pre-fetching method depends on the accuracy of the pre-analysis. Although the behavior and functions of most real-time transactions are much more predictable comparing with those in conventional database systems, the actual execution path of a real-time transaction is still difficult to determine with certainty before its execution [ORP96]. It is likely that a transaction may have several pre-defined paths for execution and which path it will follow depends on the values of its accessed data items. Since most of the MRTTs are short transactions, the total cost of pre-fetching data items required by different paths should not be expensive. However, if the MRTT are long transactions and consists of many different paths, i.e., the transaction structure shows in Figure 5, the total pre-fetching cost can be expensive.

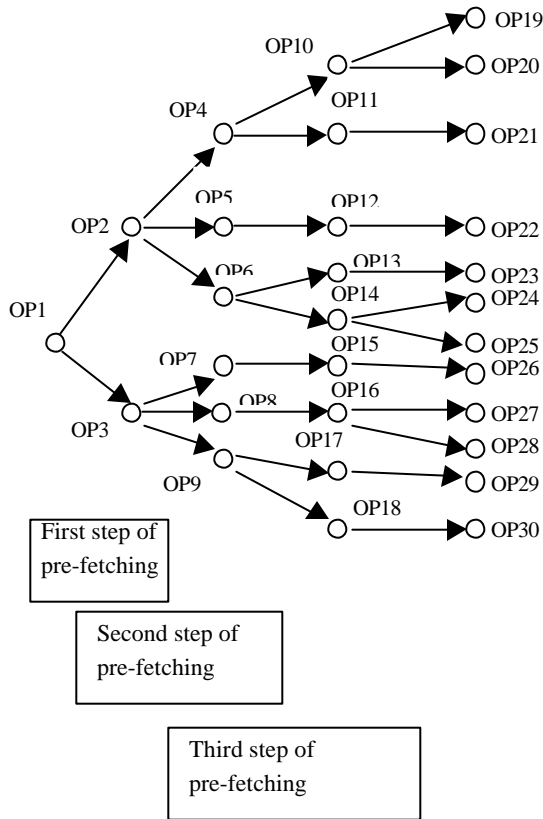


Figure 5: An example of a complex MRTT

For this case, we may perform the pre-fetching in a stepwise manner in order to maintain the total data pre-fetching workload within a low level. In the pre-analysis, pivot operations are identified. For example, in Figure 5, operations, Op1, Op2, Op3, and Op4, etc., are pivot operations. Every time after a pivot operation (or a set of operations) has been completed, the pre-fetching procedure for the next step will be performed. The size of a pre-fetching procedure can be defined in turns of the total number of data items to be pre-fetched. Note that delaying the pre-fetching of the data items required by latter operations will not affect the performance provided that the required data items are in the anchor host of the MRTT before they are required by the MRTT.

6. DISCUSSION

In a mobile environment, the performance of the system can be seriously affected by network performance especially when some of the required data items of a transaction are located at a mobile host. The issues on how to get the required data item from fixed hosts and mobile hosts, and to provide a consistent view to the transaction are challenging issues. With the multi-version data model, we relax the consistency requirement to relatively consistency. Although the data currency may be affected for some cases, i.e., under network disconnection, the probability of meeting the transaction deadline can be greatly increased. Note that when a transaction wants to access a data item, it tries to get the latest one first. The only case where it has to choose an older version is that the previous accessed data items are not relatively consistent with the latest version.

Another benefit of the multi-version data model is that it makes the cost for providing consistent data to a transaction much smaller and less affected by network disconnection. The relative consistency requirement of the data items accessed by a transaction can be ensured by checking the validity intervals of the data items.

With the image transaction to pre-fetch data items for the execution of a MRTT, the data access delay can be much reduced. When a transaction has to be restarted due to the problem of relative inconsistent, the restart overhead will not be expensive. It can find a set of relatively consistent data items at the anchor site without sending any new data requests by reading the older versions. The image sub-transactions represent the agent for the MRTT to get its required data items. It can also used to deal with the data access problem under network disconnection.

The overhead of the pre-fetching is of course the higher data transmission overheads since multiple versions of data items are being pre-fetched. The overhead will be higher when the transactions have different execution paths. However, comparing with mobile network, the network bandwidth of a wired network is much higher.

7. CONCLUSION

The research in executing real-time transactions in a mobile environment has received growing interest in recent years. Owing to the poor quality of services provided by a mobile network, it is not easy to meet transaction deadlines in a mobile environment. Most of the previous studies in the area proposed to resolve the unreliable network problem by relaxing the data consistency requirements of a transaction and by breaking a transaction into sub-transactions. Although these methods may be effective for conventional mobile database systems, they may not be suitable to a mobile real-time database system (MRTDBS), which are mostly mission critical.

In this paper, we propose to adopt relative consistency as the correctness criterion for executing real-time transactions in a mobile environment by exploring the characteristics of a mobile network, e.g., a high probability of disconnection and a low bandwidth, and the characteristics of data, e.g., a high rate of value changing, in real-time applications. Note that the relative consistency requirement defined in this paper can work together with other relaxation methods, such as epsilon serializability and similarity. We hope that, by introducing one more dimension for relaxing the transaction consistency requirement, the user

requirements and the transaction deadline requirements can be better satisfied. To accommodate the relative consistency requirements, a multi-version data model is adopted in which a set of versions is maintained for a data item, and each version has a validity interval.

Based on the pre-defined characteristics of transactions in real-time applications, we propose an image transaction model to reduce the impacts of mobile network on the system performance. An image transaction consists of a set of image sub-transactions, and each sub-transaction consists of a set of read operations. They pre-fetch the data items which will be required by the originating mobile real-time transaction. After the versions of the required data items are pre-fetched to the anchor host, where the originating transaction is initiated, a version selection algorithm will be executed. While a transaction is executing, it always reads the most recent versions of its required data items, and at the same time, ensure that all its accessed data items are relatively consistent. With such kind of data pre-fetching, the data accessing delay can be cut down tremendously. Finally, with the image transaction, the management and the structure of a real-time transaction can be much simplified since the whole transaction may only consist of a single process at the anchor host.

ACKNOWLEDGMENTS

The work described in this paper was partially supported by a grant from the Research Grants Council of Hong Kong SAR, China [Project No. CityU 1078/00E] and a research grant from Taiwan's National Science Council [NSC89-2213-E-002-200].

REFERENCE

- [AGM92] R.J. Abbott and Hector Garcia-Molina, "Scheduling Real-Time Transactions: A Performance Evaluation", *ACM Transactions on Database Systems*, vol. 17, no. 3, pp. 513-560, 1992.
- [BGS01] Ph. Bonnet, J. Gehrke, P. Seshadri, "Towards Sensor Database Systems", in *Proceedings of 2nd International Conference on Mobile Data Management*, Hong Kong, January 2001.
- [HLC90] J.R. Haritsa, M. Livny, M.J. Carey, "On Being Optimistic about Real-Time Constraints", in *Proceedings of the 9th ACM Symposium on Principles of Database Systems*, pp. 313-343, 1990.
- [HSR92] J. Huang, J. Stankovic and K. Ramamritham, "Priority Inheritance in Soft Real-Time Databases", *Journal of Real-Time Systems*, vol. 4, no. 3, pp. 243-268, 1992.
- [IB94] T. Imielinski and B. R. Badrinath, "Mobile Wireless Computing: Challenges in Data Management", *Communications of the ACM*, vol. 37, no. 10, pp. 18-28, 1994.
- [KLA99] B. Kao, Kam-Yiu Lam, B. Adelberg, R. Cheng and T. Lee, "Updates and View Maintenance in Soft Real-Time Database Systems", in *Proceedings of 1999 Conference on Information and Knowledge Management*, Kansas City, Nov 1999.
- [KM92] T.W. Kuo and A.K. Mok, "Application Semantics and Concurrency Control of Real-Time Data-Intensive Applications", in *Proceedings of IEEE 13th Real-Time Systems Symposium*, pp. 35-45, 1992.
- [KS96] Y. K. Kim, S.H. Son, "Supporting Predictability in Real-Time Database Systems", in *Proceedings of Real-Time Technology and Applications Symposium*, Brookline, Massachusetts, pp. 38-48, 1996.
- [KU99] E. Kayan and O. Ulusoy, "An Evaluation of Real-Time Transaction Management Issues in Mobile Database Systems", *The Computer Journal*, vol. 42, no. 6, pp. 501-510, 1999.
- [LC00] Yin-Bing Lin and Imrich Chlamtac, *Wireless and Mobile Network Architecture*, John Wiley and Sons, 2000.
- [LKL99] Kam-Yiu Lam, Tei-Wei Kuo, Gary C.K. Law and Wai-Hung Tsang, "A Similarity-Based Protocol for Concurrency Control in Mobile Distributed Real-Time Database Systems", in *Proceedings of International Workshop in Parallel and Distributed Real-time Systems*, Pauto Rico, April 1999.
- [LKT00] Kam-Yiu Lam, Tei-Wei Kuo, Wai-Hung Tsang and Gary C.K Law, "Concurrency Control in Mobile Distributed Real-time Database Systems", *Information Systems*, vol. 25, no. 4, June 2000, pp. 261-322.
- [ORP96] P. O'Neil, K. Ramamritham and C. Pu, "A Two-Phase Approach to Predictably Scheduling Real-time Transactions", in *Performance of Concurrency Control Mechanisms in Centralized Database Systems*, edited by V. Kumar, Prentice Hall, New Jersey, 1996.
- [OS95] G. Ozsoyoglu and R. Snodgrass, "Temporal and Real-Time Databases: A Survey", *IEEE Transactions on Knowledge and Data Engineering*, vol. 7, no. 4, pp. 513-532, 1995.
- [PB95] Pitoura, E. and Bhargava, B. , "Maintaining Consistency of Data in Mobile Distributed Environment," in *Proceedings of the 15th International Conference on Distributed Computing Systems*, pp. 404-413, 1995.
- [R93] K. Ramamritham, "Real-time Databases", *International Journal of Distributed and Parallel Databases*, vol. 1, no. 2, pp. 199-226, 1993.
- [SPL92] S.H. Son, S. Park & Y Lin, "An Integrated Real-time Locking Protocol", in *Proceedings of the International Conference on Data Engineering*, pp. 527-534, 1992.
- [U95] O. Ulusoy, "A Study of Two Transaction Processing Architectures for Distributed Real-time Database Systems", *Journal of Systems and Software*, vol. 31, no. 2, pp. 97-108, 1995.
- [U98] O. Ulusoy, "Real-Time Data Management for Mobile Computing", in *Proceedings of International Workshop on Issues and Applications of Database Technology (IADT'98)*, Berlin, Germany, pp. 233-240, 1998.
- [YWLS94] P.S. Yu, K.L. Wu, K.J. Lin and S.H. Son, "On Real-Time Databases: Concurrency Control and Scheduling", *Proceedings of IEEE*, vol. 82, no. 1, pp. 140-57, 1994.