

Transaction Shipping Approach for Mobile Distributed Real-time Databases

Kam-yiu Lam¹, Tei-Wei Kuo², Wai-Hung Tsang¹ and Gary C.K Law¹

Department of Computer Science¹
City University of Hong Kong
83 Tat Chee Avenue, Kowloon
HONG KONG
Email: cskylam@cityu.edu.hk

Department of Computer Science and
Information Engineering²
National Chung Cheng University
Chiayi, 621 Taiwan, ROC
Email: ktw@cs.ccu.edu.tw

Abstract. Due to the unpredictability of mobile network, it is difficult to meet transaction deadlines in a mobile distributed real-time database system (MDRTDBS). We propose the idea of transaction shipping to reduce the overheads in processing a transaction over mobile network and in resolving priority inversion. We consider a distributed lock-based real-time protocol, the Distributed High Priority Two Phase Locking (DHP-2PL), to study the impacts of mobile network on real-time data access. A detailed model of a MDRTDBS has been developed, and a series of simulation experiments have been performed to evaluate the performance of our approach.

1 Introduction

The realization of “instant” information access over mobile network relies on efficient real-time transaction processing techniques which are suitable to mobile environments. As a result, research in *mobile distributed real-time database systems* (MDRTDBS) is receiving growing attention in recent years [9,12,13]. However, owing to the intrinsic limitations of mobile computing systems, such as limited bandwidth and power supply for client systems, frequent disconnection, and mobility of users, the design of an efficient and cost-effective MDRTDBS requires techniques that are quite different from those adopted in distributed real-time database systems (DRTDBS) over a wired network. It is much more difficult to meet transaction deadlines in a mobile environment. There exist many factors which can seriously affect the system performance: low communication bandwidth often makes the network be the bottleneck resource. Mobility of clients also affects the distribution of workload in the entire system. Disconnection, while a transaction is processing, may even introduce the inconsistency problem. Such poor quality of mobile network can seriously affect the effectiveness of a concurrency control protocol in resolving data-access conflicts.

Although different real-time concurrency control protocols have been proposed for single-site and distributed RTDBS, they are not suitable to MDRTDBS, due to the unique characteristics of mobile environments. The poor quality of services provided by mobile network also seriously increases the overheads of a concurrency control protocol in resolving data-access conflicts. In this paper, we propose the idea of transaction shipping to process transactions issued by mobile clients. We aim at reducing the overheads for processing transactions over a mobile network and for resolving priority inversion. We consider a distributed lock-based real-time protocol, Distributed High Priority Two Phase Locking (DHP-2PL), to study the impacts of mobile network on real-time data access, where DHP-2PL is derived from the well-known High Priority Two Phase Locking (HP-2PL) [1] for MDRTDBS.

2 Model of a Mobile Distributed Real-time Database System

2.1 System Architecture

A MDRTDBS consists of four major components: the mobile clients (MC's), the base stations, the mobile network, and the main terminal switching office (MTSO) [6,7,9], as shown

in Figure 1. The mobile network is assumed to be a radio cellular network, and the entire service area is divided into a number of connected cell sites. Within each cell site, there is a base station which is augmented with a wireless interface to communicate with the MC's within its cell site. The cellular radio network is assumed to be the Global Systems for Mobile Communication (GSM) 900 in which the bandwidth is divided into a number of channels.

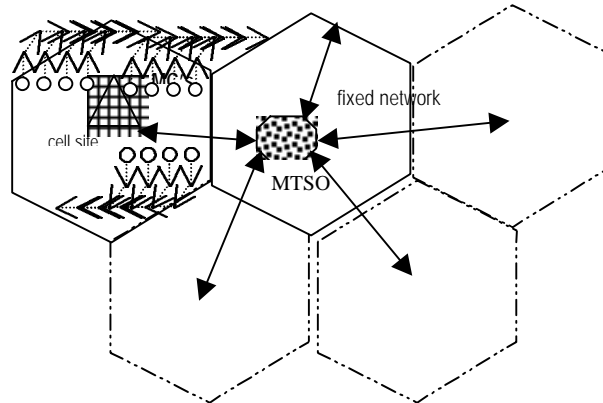


Figure 1: System Architecture of the Mobile Distributed Real-time Database System

The base stations at different cell sites are connected to the MTSO by a point-to-point wired network. Thus, the communications between the base stations and the MTSO are much more efficient than the communications between the base stations and their mobile clients. The MTSO is responsible for active call information maintenance, performance of handoff procedure, channel allocation, and message routing. Attached to each base station is a real-time database system containing a local database which may be accessed by transactions issued from MCs within its cell site or from other cells via the MTSO.

An MC may move around within the same cell site or across the cell border into another cell site. Periodically, it sends a location signal to its base station through the uplink channel. The strength of the location signal received by a base station is dependent on several factors, such as the distance between the MC and the base station, and the surrounding buildings. When an MC is crossing the cell border, the signal will become very weak. If the signal received by a base station is lower than a certain threshold level, the MTSO will be notified, and the MTSO will perform a handoff procedure. It sends out requests to all the base stations, and the base stations must respond by returning the strength of the location signals received from the MC. The MTSO will then assign the MC to the base station which has received the strongest signal. Usually, this is the base station which is responsible for the cell site which the MC is entering.

The processing of an MC transaction may need to access databases at several base stations. The MC first attempts to issue a call request to the base station of its cell site. A channel is granted to the MC after the completion of a setup procedure. The execution of the setup procedure incurs a fixed overhead as it involves communications amongst the MC, the base stations, and the MTSO. Since the number of channels between a base station and its MC's is limited, it is possible that the channel request may be refused due to no free channel available. The queuing for the channels is based on the priorities of the transactions. If the number of attempts exceeds a specified maximum number, the call and the transaction will be aborted. Due to channel contention and slow (and unreliable) communication, the time required to establish a channel is highly unpredictable. Once a channel has been established, the transaction will be sent out through the RF transmitter, which is connected to the antenna from the MC, to the base station of the cell site where the MC currently resides. When an MC is crossing the cell site border while it is communicating with its base station, a new channel will be created

with its newly assigned base station after a setup procedure. However, it is still possible that there is no channel available at the new cell. In this case, it will retry for a specified number of times. If it is still not able to get a channel, the transaction will be aborted. Due to noise and interference, the signal, which carries data, may be corrupted while it is being transmitted. In this case, the data will be re-transmitted. Because of high error rate and non-stability of signal transmission, the effective data transmission rate over a mobile network is also highly unpredictable.

2.2 Database and Transaction Models

The entire database is partitioned over different base stations. At each base station, there is a local real-time database, which consists of two types of data objects: *temporal* and *non-temporal* data objects. Temporal data objects are used to record the status of the external objects in the real world. Each temporal data object is associated with a timestamp which denotes the age of the data object. If a transaction may update a temporal data object, then the transaction is given a timestamp when it is initiated. If the transaction commits successfully before its deadline, then the timestamp of any data object which is updated by the transaction will be set as the timestamp of the transaction. The validity of a temporal data object is defined by an *absolute validity interval (avi)* [11]. A temporal data object satisfies the *avi* constraint if the age of the data object is up to date, i.e., the difference of the current time and the age is no more than *avi*. A *relative validity interval (rvi)* may be given to a transaction which requires that the maximum age difference of data objects read by the transaction is no larger than *rvi*. Non-temporal data objects are either derived by operations of transactions or are statically set during system initialization.

Each transaction is given a deadline and a criticality. The priority of a transaction is derived based on its deadline and criticality. It is assumed that the EDF scheduling is used for scheduling the transactions in using the CPU at the base station. It is assumed that the transactions are firm real-time [10]. If the system cannot complete a transaction before its deadline, the transaction will be aborted. Each transaction consists of a sequence of operations. Each operation requires the access of one to several data objects. Operations may access data objects residing at several base stations. When an operation of a transaction accesses a data object residing at a base station outside the cell site of the MC, the operation will be routed to the base station via the MTSO. When all of the operations of a transaction have been processed, a commit protocol will be performed to ensure the failure atomicity of the transaction. It is assumed that the well-known two phase commit (2PC) is adopted because of its simplicity.

3 A Distributed Real-time Locking Protocol for MDRTDBS

In this paper, we propose a distributed locking approach based on both restart and priority inheritance to resolve the problem of priority inversion in MDRTDBS. By extending the well-known HP-2PL, a distributed extension of HP-2PL, called Distributed HP-2PL (DHP-2PL), is proposed. In the design of DHP-2PL, we shall reduce the number of transaction restarts and resolve lock conflict between higher priority and lower priority transactions globally or locally because transaction restarts and conflict resolution can be very expensive in a mobile environment. Unlike other blocking-based protocols such as those based on priority inheritance, DHP-2PL is free of deadlock because priority inheritance is used restrictively for resolving lock conflict with committing transactions in the DHP-2PL.

It is assumed that the local database system at each base station has a lock scheduler which handles the lock requests for data objects residing at that base station. The definition of DHP-2PL is as follows, where T_r and T_h are the lock requesting and lock holding transactions, respectively:

```

Lock Conflcit (Tr, Th)
Begin
  If      Priority(Tr) > Priority(Th)
  If      Th is not committing or restarting
  If      Th is a local transaction
          Restart Th locally
  Else
          Restart Th globally
  Endif
  Else
    Block Tr until Th releases the lock
    Priority(Th) := Priority(Tr) + fixed
    priority level
  Endif
  Else
    Block Tr until Th releases the lock
  Endif
End

```

A transaction is local if it only has accessed data objects residing at one base station. Otherwise, it is a global transaction. Similar to the HP-2PL, the DHP-2PL uses a restart approach to resolve lock conflicts between non-committing transactions. Restarting a local transaction is simply done by restarting the transaction at the conflicting base station. To restart a global transaction, restart messages must be sent to the base stations where some operations of the global transactions are executing. Global restart takes a much longer time and requires much higher overhead. Usually, a global transaction takes more resources for execution. Restarting a global transaction implies that a lot of resources will be wasted. Thus, it should be minimized. So, if a global transaction is committing, it is allowed to hold a lock until it has finished the commit procedure, even though a higher priority transaction is requesting the lock. Although this will create the priority inversion problem, the blocking time of the higher priority transaction is minimized by priority inheritance. In the DHP-2PL, the priority of the committing transaction will be raised up by two factors: Firstly, its priority is no less than the highest priority of all its blocked transactions. Secondly, a fixed priority level is added to its priority to let its priority higher than all the other executing transactions. No deadlock is possible for the priority raising of any committing transaction because the committing transaction will not be blocked by any other executing transactions. It is because a committing transaction will not make any lock request during its commit procedure.

4 Transaction Shipping for MDRTDBS

To improve the system performance and to reduce the impact of a mobile network on the performance of the DHP-2PL in a MDRTDBS, we propose a *transaction shipping* approach to process the transactions from the mobile clients. The purpose of the approach is to reduce the amount of communications between the mobile client and the database server to alleviate the dependency of the system performance on the underlying network in a MDRTDBS.

In the transaction shipping approach, the entire transaction will be “shipped” to the server for processing instead of shipping every operation or data request to the database server. Although the idea is simple, there exist several difficult practical issues needed to be solved, such as how to identify the path of a transaction before its execution, how to define the execution model of a transaction in a mobile environment, and how to deal with the dynamic properties in transaction execution. For some mobile applications, they may even require users to input parameters while the transactions are executing in order to determine the path of execution. The problem is further complicated by the limited processing power and memory sizes of the mobile clients. Due to these limitations, the client structure is usually very simple

with limited information about the whole system e.g., the location of the data objects. In order to solve the problem in transaction shipping, we suggest a pre-analysis approach. The practicality of the pre-analysis comes from the fact that the behavior of many real-time transactions is more predictable, compared with other conventional database systems.

4.1 Pre-analysis Phase

In the transaction shipping approach, the execution of a transaction is divided into two phases: the *pre-analysis phase* and the *execution phase*. Figure 2 shows the system architecture of the processing of a transaction under the transaction shipping approach. Once a transaction is initiated at a mobile client, a coordinator process, called *master coordinator*, will be created at the client side. Before shipping a transaction to the database server, the system will perform a pre-analysis on the transaction to derive its characteristics, e.g., what the operations of the transaction are and what the execution path of the transaction is. The concept of pre-analysis is similar to the two phase methods discussed in [5]. However, it should be noted that [5] is concerned about how to reduce the unpredictability in data access by using the concept of access invariant. In here, we use the pre-analysis to predefine the execution path of a transaction in order to reduce the number of communications between mobile clients and the base stations.

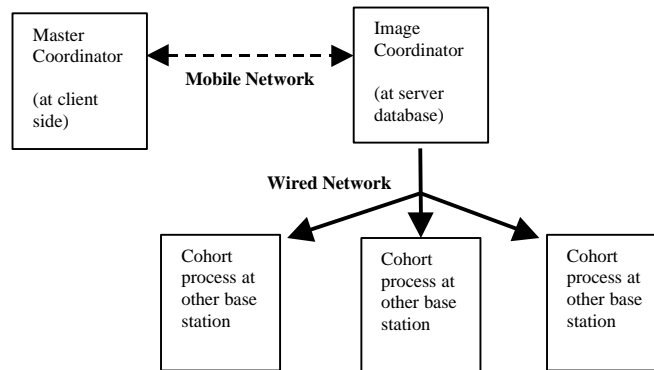


Figure 2: Process Architecture under Transaction Shipping Approach

Although transaction pre-analysis is considered to be difficult and impractical in many conventional database systems due to their dynamic nature, it is quite feasible for many real-time database applications [4,5] *as the properties and behavior of transactions in many real-time database applications are much more well-defined*. Furthermore, in a real-time database system, transactions can often be classified into different types in which each transaction type has certain pre-defined characteristics, e.g., the set of operations and their required data objects, for performing specific user operations [4]. For such real-time database applications, partial pre-analysis can even be done in an *off-line fashion*.

The pre-analysis is started at the mobile client upon the generation of the transaction. It consists of two phases. In the first phase, the *set of operations* in the transaction will be identified. It is usually not difficult to identify the operations in a transaction. For example, if SQL statements are used to access the database, SELECT and INSERT statements are read and write operations, respectively. Note that transactions are assumed to have a simple flat structure. At this stage, it may not be necessary to identify the set of data objects required by the operations. Actually, it may not be easy to be done at the client side as it only contains limited information about the database system and the location of the data objects in the

system. The required data objects of an operation can be determined when the operation is being processed at the base station.

In the second phase, the *execution path* of the transaction, e.g., the precedence relationships of its operations, will be determined. For some transactions, the whole execution path can not be determined until the data objects required by the transactions have been identified. For such transactions, the pre-analysis may identify the transaction type first and then, based on the pre-defined characteristics of that transaction type, to make the prediction. After the completion of the pre-analysis, a signature of the transaction will be created in which the operations of the transaction and their precedence relationships are defined.

4.2 Execution Phase

The signature transaction is forwarded to the base station of the MC through the mobile network. Once the server receives the transaction signature, it will create a process, called *image coordinator*, for the transaction. The image coordinator will take over the job from the master coordinator at the MC to process the transaction. Other cohorts for the executions of the operations (issued by the transaction) will be created at other base stations if the operations need to access the data objects located at that base stations.

The benefit of defining an image coordinator at the base station instead at the MC is that the connections amongst the base stations are much better than the connection between the MCs and its base stations. This approach can facilitate the management of transactions and improve the performance of the atomic commitment protocol. Whenever a transaction has to be restarted, all its cohort processes (excluding the master coordinator and the image coordinator) will be destroyed after the completion of undo operations. The image coordinator will be responsible for restarting the transaction from its beginning if its deadline has not been expired.

4.3 Dynamic Properties of Transactions

Due to the dynamic nature of transactions (although transactions in real-time database applications are more well-defined, compared with transactions in traditional database systems, they may still have some dynamic properties) and the interactivities between transactions and mobile clients, data input from MC users may still be needed. As a result, the pre-analysis of a transaction may need to be re-done while it is executing. In this case, the transaction may have to go back to its MC after the completion of an operation. It is obvious that the performance gained from the pre-analysis is dependent on the accuracy of the pre-analysis. In the best case, one single communication between the MC and the base station is required for each transaction execution. In the worst case, the number of communications between the mobile client and the base station for each transaction execution is equal to the number of operations in the transaction, which is equal to that required in the query shipping approach. If the system can make a good estimation in the pre-analysis, a lot of communication cost can be saved.

Note that the transaction shipping approach does not require the transmission of a large number of data objects to the clients, as required in the data shipping approach. This is a very attractive feature for MDRTDBS. Although the processing of the entire transaction at the database server may increase the workload at the base stations, it can ease the management of transactions because all of the cohorts for the execution of a transaction are located at the base stations connected by wired network through the MTSO. Also, the processing power of a mobile client is usually much lower than the servers at the base stations. A mobile client is more suitable for performing simple jobs, e.g., the pre-analysis. The actual processing of the transactions is performed at the base stations which are much more powerful and stable, and the base stations contain much more information about the system.

5 Performance Experiments

In order to evaluate impact of mobile communication on the performance of the proposed protocol DHP-2PL and the transaction shipping approach, a detailed simulation program of a MDRTDBS is implemented according to the MDRTDBS model defined in Section 2.

5.1 Model Parameters and Performance Measures

The deadline of a transaction, T , is defined according to the expected execution time of a transaction [1,2,3,8,9] such as:

$$Deadline = ar(T) + pex(T) \times (1 + SF)$$

where SF : the slack factor is a random variable uniformly chosen from a slack range;
 $ar(T)$: the arrival time of transaction T ;

$pex(T)$: the predicted execution time of T . It is defined as:

$$pex(T) = (T_{lock} + T_{process} + T_{update}) \cdot N_{oper}$$

where N_{oper} : the number of operations in a transaction;

T_{lock} : the CPU time required to set a lock;

$T_{process}$: the CPU time required to process an operation; and

T_{update} : the CPU time to update a data object (for write operations).

The baseline setting of the model is shown as follows:-

Parameters	Baseline Values
Number of MTSO	1
Number of Cell Sites	7
Location Update Interval	0.2 second
Transmission Speed for Channel	10 kbps
Number of Channels for Each Cell Site	10
Think Time	8 seconds
Transaction Size	7 to 14 operations (uniform distribution)
Proportion of write operations	1.0
Slack range	10 – 20 (slack factor is uniformly distributed in the slack range)
Number of Mobile Clients	84
Channel Connection time (CL)	1 second
Call Update Interval	0.2 second
Number of Local Databases	7 (1 in each base station)
Database Size	200 data objects per local database
Concurrency Control	Distributed High Priority Two Phase Locking (DHP-2PL)
Fraction of Temporal Data Objects	10%
Temporal Data Object Update Interval	0.5 update per second per data object
Absolute Threshold (avi)	12 seconds
Relative Threshold (rvi)	8 seconds
CPU Scheduling	Earliest Deadline First
Deadline Missing Treatment	Firm deadline, abort the transaction once the transaction deadline is found missing

Table 1: Model parameters and their baseline values

The primary performance measure used in the simulation experiments is the *miss rate* which is defined as the number of transactions which missed deadlines over the total number of transactions generated. In addition to miss rate, we also measure the *restart probability* which

is defined as the total number of transaction restarts over the number of committed transactions. The restart probability can be used as an indicator of data conflict probability. Other measures are *call blocking probability*, *mean call waiting time*, *mean turn around time* of a transaction and the *CPU utilization*. The call blocking probability is defined as the number of failed channel requests over the total number of channel requests.

5.2 Performance Results and Discussion

In this section, we study the performance of the transaction shipping approach (TS) as compared with the query shipping approach (QS). We have tested the performance of the transaction shipping approach under different workload overhead and return probability. Under the transaction shipping approach, a transaction has to go back to the mobile client when: (1) it has to receive input data from the mobile client; or (2) when the prediction done at the pre-analysis is incorrect. For the later case, the prediction has to be performed again at the client side. The return probability defines the probability of a transaction to go back to the mobile client after the completion of an operation in the transaction shipping approach. Its value depends on the accuracy of the prediction done at the pre-analysis.

Figure 3 shows the miss rate under the transaction shipping approach (TS) at different return probability and at a zero overhead for the pre-analysis as compared with the query shipping approach (QS). (The impact of the overhead for pre-analysis will be studied in the next set of experiments.) It can be seen that the performance of the system is greatly improved with the use of the transaction shipping approach especially when the return probability is low (a high accuracy of prediction in the pre-analysis). When the return probability is smaller than 0.6, the improvement is more than 75% when the channel connection time is 1 second ($CL = 1$) and the improvement is about 50% when the channel connection time is 2 second ($CL = 2$). Again, this is due to smaller channel blocking probability and call waiting time as can be seen in Figure 4 and Figure 5, respectively. The use of transaction shipping approach greatly reduces the amount of communication between the mobile clients and the base stations required for processing the transactions. This also makes the mean turn around time of transactions much shorter as shown in Figure 6. Due to the lower channel contention, part of the workload is shifted from the mobile network to the CPU when the return probability is low (as shown in Figure 7). However, when the return probability and the channel connection time are high, the network will still be the bottleneck resource.

The use of the transaction shipping approach also reduces the lock conflict probability as the lock holding time by a transaction is shorter. The smaller conflict probability can be observed in Figure 8. The restart probability drops greatly with a decrease in return probability when the channel connection time is 1 second. However, when the channel connection time is long ($CL = 2$), the restart probability drops slightly only when the return probability is very high. It can be explained as follows. For a high channel contention due to a larger return probability and longer channel connection time, some transactions may miss their deadlines even before they have started the processing. Thus, the number of lock holding transactions will be smaller and the probability of lock conflict will be lower. Since the number of restart with the transaction shipping approach is much smaller, the degree of channel contention is lower.

Figure 9 shows the impact of different overheads (defined in terms of the amount of time) for the pre-analysis in the transaction shipping approach on the miss rate. It can be observed that the overhead does not have any observable effect on the performance of the transaction shipping approach. Although the overhead will make the deadlines more tight as the time required to complete a transaction becomes longer, it also releases the degree of resource contention in the system (the mobile network and the base stations) especially on the channels as the transactions now spent more time at the mobile client side instead at the mobile network and base stations.

6 Conclusions

Due to the poor quality of services provided by a mobile network, it is not easy to meet the deadlines of transactions in a mobile distributed real-time database system (MDRTDBS). The performance of any adopted concurrency control protocol in MDRTDBS will be very different from its performance in a DRTDBS over wired network. In this paper, we propose the idea of transaction shipping to process transactions issued by mobile clients. We aim at reducing the overheads for processing a transaction over mobile network and for resolving priority inversion. We consider a distributed lock-based real-time protocol, the Distributed High Priority Two Phase Locking (DHP-2PL), to study the impacts of mobile network on real-time data access, where DHP-2PL is derived from the well-known High Priority Two Phase Locking (HP-2PL) [1] for MDRTDBS. Extensive simulation experiments have been conducted to investigate the impact of mobile network on the performance of the DHP-2PL protocol. Since bandwidth is one of the most scarce resources in a mobile environment, the study shows that the call duration has a significant impact on the performance of DHP-2PL. With the transaction shipping approach, the number of deadline violations is greatly reduced as the contention for channel, the time spent on communication, the probability of lock conflict, and the amount of resources wasted on restarted transactions are much reduced.

References

- [1] R. Abbott and Hector Garcia-Molina, "Scheduling Real-Time Transactions: A Performance Evaluation," *ACM Trans. on Database Systems*, vol. 17, no. 3, pp. 513-560, Sep. 1992.
- [2] J.R. Haritsa, M. Livny, M.J. Carey, "On Being Optimistic about Real-Time Constraints", in the *Proceedings of the 9th ACM Symposium on Principles of Database Systems*, 1990.
- [3] J. Huang, J. Stankovic & K. Ramamritham, "Priority Inheritance in Soft Real-Time Databases", *Journal of Real-Time Systems*, vol. 4, no. 3, pp. 243-268, 1992.
- [4] Y.K. Kim, S.H. Son, "Supporting Predictability in Real-Time Database Systems", *Proceedings of Real-Time Technology and Applications Symposium*, Brookline, Massachusetts, June 1996, pp. 38-48.
- [5] P. O'Neil, K. Ramamritham & 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.
- [6] OPNET Modeler/Radio 3.0.B ©, MIL 3, Inc., 1996.
- [7] E. Pitoura and B. Bhargava, "Dealing with Mobility: Issues and Research Challenges," Technical Report, Purdue Univ., Nov. 1993.
- [8] 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.
- [9] O. Ulusoy, "Real-Time Data Management for Mobile Computing", *Proceedings of International Workshop on Issues and Applications of Database Technology (IADT'98)*, Berlin, Germany, July 1998.
- [10] P.S. Yu, K.L. Wu, K.J. Lin & S.H.Son, "On Real-Time Databases: Concurrency Control and Scheduling," *Proceedings of IEEE*, vol. 82, no. 1, pp. 140-57, 1994.
- [11] M. Xiong, K. Ramamritham, R. Sivasankaran, J.A. Stankovic, and D. Towsley, "Scheduling Transactions with Temporal Constraints: Exploiting Data Semantics," *Proceedings of IEEE Real-Time Systems Symposium*, December 1996, pp. 240-251.
- [12] P. Xuan, O. Gonzalez, J. Fernandez & K. Ramamritham, "Broadcast on Demand: Efficient and Timely Dissemination of Data in Mobile Environments", *Proceedings of 3rd IEEE Real-Time Technology Application Symposium*, 1997.
- [13] Ersan Kayan and Ozgur Ulusoy, "Real-Time Transaction Management in Mobile Computing Systems" in *Proceedings of 6th International Conference on Database Systems for Advanced Applications*, Taiwan, April 1999.

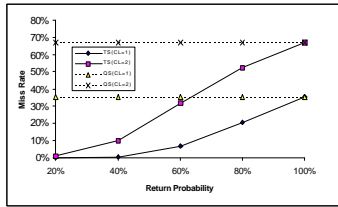


Figure 3 Miss rate at different return probabilities

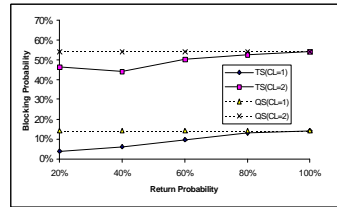


Figure 4 Blocking probabilities at different return probabilities

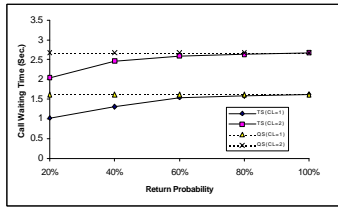


Figure 5 Call waiting time at different return probabilities

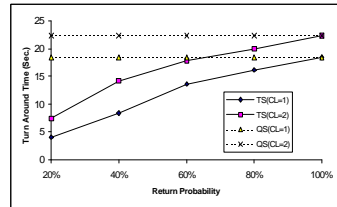


Figure 6 Turn around time at different return probabilities

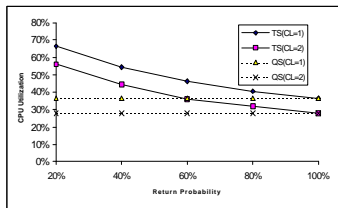


Figure 7 CPU utilization at different return probabilities

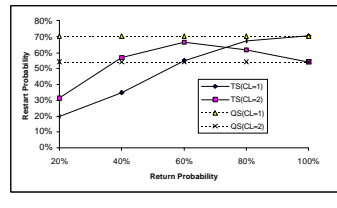


Figure 8 Restart probabilities at different return probabilities

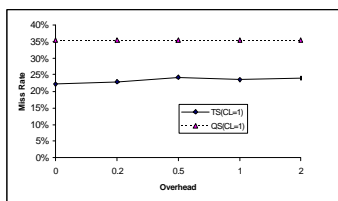


Figure 9 Miss rate at different overhead for pre-processing