An Evaluation of Real-Time Transaction Management Issues in Mobile Database Systems

Ersan Kayan and Özgür Ulusoy

Department of Computer Engineering and Information Science
Bilkent University
Bilkent, Ankara 06533, Turkey

Abstract

A critical issue in mobile data management is to respond to real-time data access requirements of the supported application. However, it is difficult to handle real-time constraints in a mobile computing environment due to the physical constraints imposed by the mobile computer hardware and the wireless network technology. In this paper, we present a mobile database system model that takes into account the timing requirements of applications supported by mobile computing systems. We provide a transaction execution model with two alternative execution strategies for mobile transactions and evaluate the performance of the system considering various mobile system characteristics, such as the number of mobile hosts in the system, the handoff process, disconnection, coordinator site relocation, and wireless link failure. Performance results are provided in terms of the fraction of real-time requirements that are satisfied.

1 Introduction

A *mobile computing system* is a dynamic type of traditional distributed systems where the links between *mobile computers* and *mobile support stations* that provide wireless interface to mobile computers can change dynamically [5]. In a mobile computing system, users carrying mobile (portable) computers can access database services from any location without requiring to maintain a fixed position in the network. Some examples of applications supported by database systems in mobile computing include location dependent queries, long-lived transactions that require migration of data into the mobile computers, form-based transactions (e.g., inventory orders, sales with credit cards), and online information services (e.g., a local yellow pages, stock market information, banking, travel information) [5, 7].

In a mobile computing environment, the need for a real-time database management system (DBMS) is strong, because one of the basic requirements in mobile data management is to provide real-time response to transactions of the underlying application. Therefore, a critical item on the agenda for mobile data management research should be to support mobile computing systems with real-time DBMSs. The resource constraints of mobile computing systems make it difficult to satisfy timing requirements of supported applications. Low bandwidth, unreliable wireless links, and frequent disconnections increase the overhead of communication between mobile hosts and the static part of the system. Mobility of hosts makes the location information dynamic which results in a considerable increase in the cost of search and updates on mobile host location. We believe that the research results obtained in the area of real-time database systems can be adopted to mobile computing systems to obtain a reasonable real-time performance in the presence of the constraints of mobility [15].

In this paper, we present a mobile database system model that takes into account the timing requirements of applications supported by mobile computing systems. A mobile transaction execution model constructed for that system is implemented by a simulation program, and the performance of the system is analyzed to evaluate the impact of various mobile system characteristics, such as the number of mobile hosts in the system, the handoff process[1], disconnection, coordinator site relocation, and wireless link failure. Performance results are provided in terms of the fraction of real-time requirements that are satisfied.

The organization of the paper is as follows. In Section 2, we discuss the recent work on mobile transaction management. In Section 3, the mobile real-time database system model used in evaluating various transaction management issues is described. Details of the simulation program are discussed in Section 4. Results of the performance experiments are presented in Section 5. Finally, the conclusions are provided in the last section.

[1]Performing the necessary changes in configuration information when the connectivity of a mobile computer is switched from one mobile support station to another.

2 Related Work

A possible execution strategy for mobile transactions is discussed in [17]. A transaction is submitted by a mobile host to its mobile support station to be executed on the fixed part of the system, and then the result of the execution is returned back to the mobile host. Following the submission of its transaction, a mobile host may disconnect itself from the network to perform some other task. This approach does not support interactions between a mobile host and a transaction generated at that host (e.g., the transaction cannot get input from the user during its execution).

A mobile transaction execution model that supports the so called *weak operations* is proposed in [10, 11]. In this model, the database is divided into clusters of semantically related or closely located data. All data items inside a cluster are required to be fully consistent. But, degrees of inconsistency are allowed among data at different clusters. When the lack of strict consistency can be tolerated by the semantics of an application, in order to maximize local processing in a cluster and to reduce network access, the user is allowed to interact with locally available consistent data. To achieve this, *weak read* and *weak write* operations that operate on locally consistent data are introduced. Standard read and write operations are called *strict read* and *strict write* operations. Locally consistent data are accessed by issuing weak transactions that consist only of weak read and weak write operations, and globally consistent data are accessed by issuing strict transactions that consist only of strict read and strict write operations.

The mobile transaction model proposed in [3] is based on the *open-nested transaction model* and it supports *reporting transactions* and *co-transactions* to minimize communication and maintenance costs by relocating transactions. A mobile transaction consists of a set of relatively independent component transactions which can interleave in any way with other mobile transactions. A reporting transaction is a component transaction of a mobile transaction T, and it can share its partial results with T while in execution. A co-transaction is a reporting transaction in which the control is passed from one transaction to another at the time of sharing of the partial results. Reporting transactions and co-transactions can relocate their execution from one host to another.

In [16], an approach for transaction processing in mobile database systems that uses *object semantic information* is proposed. Mobile transaction processing is viewed as a concurrency and cache coherence problem. The disconnection state of a mobile host is supported by utilizing object semantic information to provide finer granularity of caching and concurrency control, and to allow for asynchronous manipulation of the cached objects on the mobile host.

3 A Mobile Real-Time Database System Model

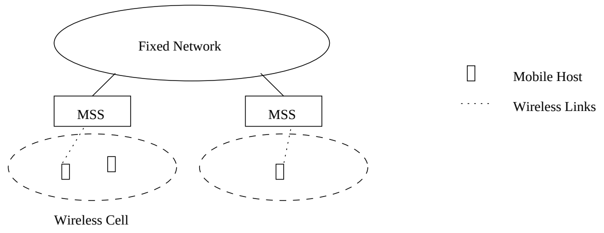

Figure 1: A Mobile Computing System

A typical mobile computing system consists of a number of mobile and fixed hosts (Figure 1). *Fixed hosts* (FHs) are connected with each other via a fixed high-speed wired network and constitute the fixed part of the system. A *mobile host* (MH) is capable of connecting to the fixed network via a wireless link. Some of the fixed hosts, called *mobile support stations* (MSSs), are augmented with a wireless interface to communicate with mobile hosts. The links between mobile computers and the MSSs can change dynamically. The geographical region in which mobile hosts can communicate with an MSS is called the *cell* of that MSS. A mobile host is local to an MSS if it is inside the cell of the MSS.

An MSS acts as an interface between the local mobile hosts and the fixed part of the network, and is responsible for forwarding messages and data between the local mobile hosts and the fixed network.

The location of a mobile host can be regarded as a data item that changes as the user crosses the boundary between two cells. Location servers at the fixed hosts are responsible for maintaining the location database and answering location-dependent queries. In order to communicate with a mobile host, it is required to find out its current location. In our mobile system model, *forwarding pointers mechanism* is used to find out the location of a mobile computer. In this mechanism, when a mobile host changes its location, its new address is deposited at the old location so that messages sent to the mobile host can be forwarded to the new address [6].

In our model, we assume that all fixed hosts act as mobile support stations (MSSs). Each MSS has a database server which enforces strict data consistency. Each mobile host is associated with a *coordinator MSS* that coordinates the operations of the transactions submitted by that mobile host. Unless otherwise stated, we will assume that the coordinator site of a mobile host is fixed.

3.1 Transaction Execution

Two types of transactions can coexist in our system: *mobile transactions* and *fixed transactions*. A mobile transaction is generated by a mobile host and can be executed at its generating host and/or some fixed hosts. A fixed transaction is generated by a fixed host and can be executed at some of the fixed hosts. Each transaction is associated with a timing constraint in terms of a *deadline*.

Our transaction execution model is an extension of the model in [14] to mobile computing systems. Each mobile transaction exists in the system in the form of a *mobile master process* (MMP) at the generating mobile host, a *fixed master process* (FMP) at the coordinator MSS of the generating host, and *fixed cohort processes* (FCPs) at sites where the required data pages reside. A fixed transaction, on the other hand, is associated with an FMP at its generating host and FCPs at various sites where it has to access data. Each transaction can have at most one cohort process at a site.

A mobile transaction consists of *Read, Write* and *User Interaction* operations while a fixed transaction consists of only *Read* and *Write* operations. Data request messages for *Read* and *Write* operations are sent to cohort processes at relevant data sites under the coordination of FMP. For each *Read* or *Write* operation a global data dictionary is referred to find out the relevant data site. Each data site has a copy of this global data dictionary. We assume that the write set of a transaction is a subset of its read set. Data pages are accessed randomly by a transaction. A *User Interaction* operation is handled at the generating host of the mobile transaction. It can be considered as reading a local data item from the generating mobile host.

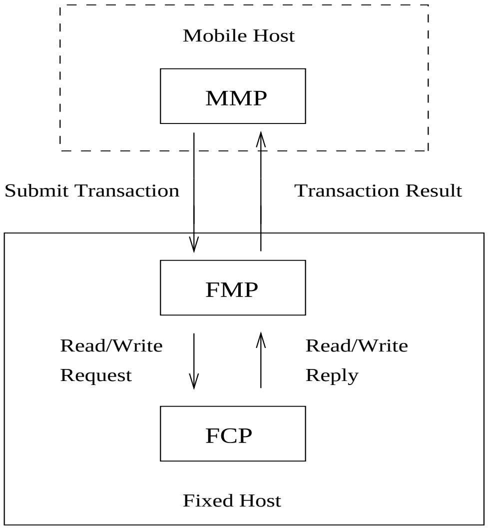

Figure 2: Mobile Transaction Execution Strategy ESFH

A mobile transaction is executed in either of the following two ways:

1. The entire transaction is submitted in a single request message to the fixed network (Figure 2). In this execution strategy, FMP has the control of the execution of the transaction. After FMP completes the execution of the transaction, it returns the result to the MMP. We call this execution strategy *ESFH (Execution Site is a Fixed Host)*.

2. A data request message for each *Read* and *Write* operation of a transaction is submitted by MMP to FMP (Figure 3). FMP handles this request in the fixed network and provides the data to MMP. Processing of each operation is performed at the mobile host. This execution strategy is called *ESMH (Execution Site is a Mobile Host)*.

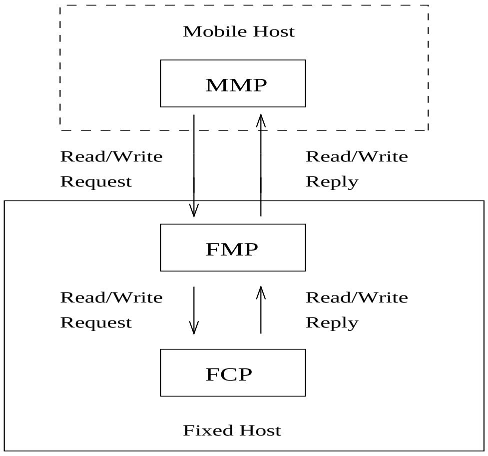

Figure 3: Mobile Transaction Execution Strategy ESMH

In the first approach, CPU power of the mobile host is not used for processing transaction operations; therefore this approach seems to be more suitable for mobile hosts which do not have powerful CPUs.

The execution of a fixed transaction is controlled completely by the FMP of the transaction at its generating host.

Each mobile/fixed transaction has a unique priority based on its timing constraint which is used in ordering resource and data access requests of transactions. Transactions are assumed to carry firm deadlines (i.e., the transaction that has missed its deadline is aborted and discarded from the system [9, 12]). We assume that the system has no apriori knowledge of the transaction execution requirements, such as the pages to be accessed and the execution time estimation of the transaction.

If a transaction aborts due to a data conflict (rather than the expiration of its deadline), it is restarted with the same deadline and priority.

3.2 Concurrency Control

Serializability is ensured through the adaptation of the two-phase locking scheme. *Priority Abort* protocol is used for resolving data conflicts (i.e., a low priority transaction is aborted when one of its locks is requested in a conflicting mode by a high priority transaction) [1]. Each fixed site in the system has a scheduler to manage the lock requests of the cohort processes executing at that site. Each cohort process has to obtain a shared lock on each data item it reads, and an exclusive lock on each data item it writes.

Global serializability is ensured by using the strict two phase locking rule; i.e., by holding the locks of a transaction until after its commitment.

3.3 Atomic Commitment

Atomic commitment of a transaction is provided by using a modified version of *Two Phase Commit* (2PC) protocol [2]. This modified version, which is described in [8], adapts 2PC to a mobile computing environment. In this protocol, FMP is designated as the coordinator, and each cohort process executing on a data site acts as a participant. When a transaction commits, the updates of its cohort processes are stored in the local databases.

3.4 Handoff

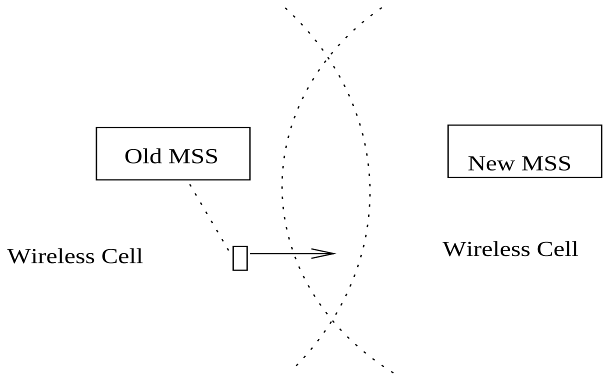

Figure 4: Handoff Operation

Due to mobility of users, mobile hosts may cross the boundary between any two cells (Figure 4). In order to keep the connectivity of a mobile host to the fixed network a *handoff* process is implemented. During handoff, the new cell's MSS takes the responsibility of providing a wireless interface to the mobile host. This process should be transparent to the user.

Our handoff protocol is similar to the one described in [4]. We assume that each MSS broadcasts *beacons* over its wireless link. Each beacon carries its sender MSS's address. A mobile host monitors the wireless signal strength it receives from neighboring MSSs. The mobile host may decide to initiate a handoff process when the signal received from a new MSS is substantially stronger than that received from the current one.

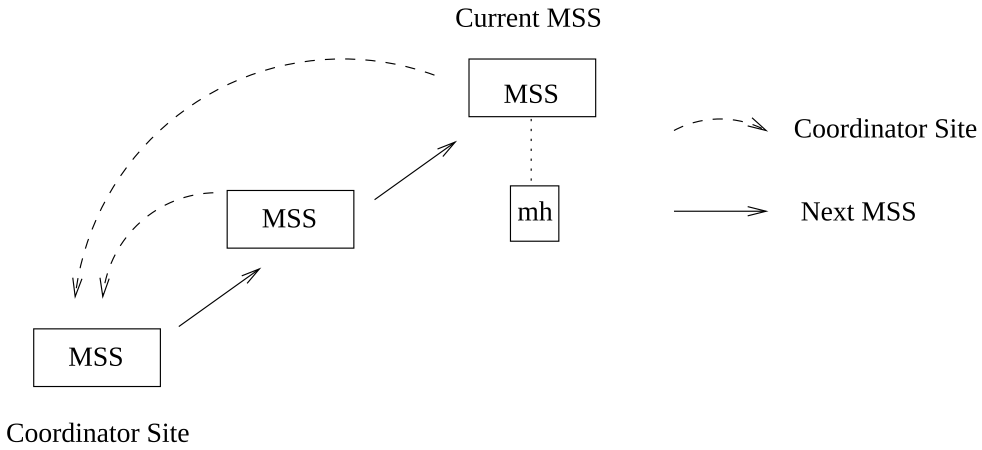

Figure 5: Linked List Structure Between Coordinator Site and Current MSS

As shown in Figure 5, a distributed linked list structure is constructed for the location information of the mobile host between the coordinator site and the current MSS of the mobile host during the handoff process.

3.5 Disconnection

One of the characteristics of mobile computing systems is frequent disconnection of mobile hosts due to the limitations on battery power and wireless bandwidth. During the disconnection period, communication between FMP and MMP of a mobile transaction is not possible. In order to prevent a mobile transaction to block other transactions due to a data conflict after its deadline expires, its FMP has the right to unilaterally abort the transaction even if it does not have the execution control of the transaction. If, on the other hand, FMP has the control of the transaction execution, it is allowed to commit or abort the transaction during disconnection. Upon reconnection of the mobile host, it informs MMP about transaction abort or commit.

4 Simulation Model

We have implemented our mobile real-time database system model on a simulation program written in CSIM [13]. The simulation model parameters are listed in Table 1. Our mobile computing system consists of *NumFHosts* fixed hosts, *NumMHosts* mobile hosts and a fixed network connecting fixed hosts. All fixed hosts are assumed to be MSSs and mobile hosts are connected to the fixed part via wireless links over the MSSs. We assume that there is no contention for wireless link bandwidth.

A fixed host has *NumFhCPU* CPUs and a disk. These resources are shared by all users. A mobile host has only one CPU and this CPU is used only by the mobile user himself. Each fixed host has a local database of size *LocalDBSize* pages. *MemSize* is the memory size of a

ExecStrategy	execution strategy
NumFHosts	number of fixed hosts
NumMHosts	number of mobile hosts
ThinkTime	think time
LocalDBSize	local database size in pages
PageSize	page size in bytes
MemSize	memory size in pages at each FH
NumFhCPU	number of CPUs at a FH
PageCPUTime	CPU time to process a data page at a FH
MsgCPUTime	CPU time to process a message at a FH
CPURatio	relative power of a FH CPU to a MH CPU
NumAccessed	number of pages accessed by a transaction
NumUserInt	number of user interactions during the execution of a mobile transaction
DiskTime	IO time to access a page on disk
UpdTrProb	fraction of update transactions
WriteProb	page write probability for an update transaction
SlackRate	slack rate
WiredBand	wired link bandwidth
WirelessBand	wireless link bandwidth
ContMsgSize	control message size in bytes
HandoffInt	mean handoff interval
HandoffProb	handoff probability
ConnectInt	mean connectivity interval
DisconProb	disconnection probability

Table 1: Simulation Parameters

fixed host in number of data pages.

The number of *Read* and *User Interaction* operations of a transaction are specified by the parameters *NumAccessed* and *NumUserInt*, respectively. A transaction is specified as an update transaction with probability *UpdTrProb*. A page that is accessed by an update transaction is updated with probability *WriteProb*. Each transaction is associated with a priority to be used in resolving data conflicts and scheduling hardware resources. The priority assignment policy used in our model is *earliest deadline first* (i.e., the transaction with the earliest deadline is assigned the highest priority).

Transactions are submitted by each host one after another. After a transaction has committed, the generating host of the transaction waits for *ThinkTime* seconds to submit the next transaction.

The features of mobility are simulated by using the parameters *HandoffInt*, *HandoffProb*, *ConnectInt* and *DisconProb*. The MSS of a mobile host can change at the beginning of each *HandoffInt* time interval with probability *HandoffProb*. A mobile host stays connected/disconnected on the average *ConnectInt* time interval, and a connected mobile host disconnects at the beginning of the next *ConnectInt* time interval with the probability *DisconProb*, and a disconnected mobile host reconnects at the beginning of the next *ConnectInt* time interval with the probability (1 - *DisconProb*).

4.1 System Components

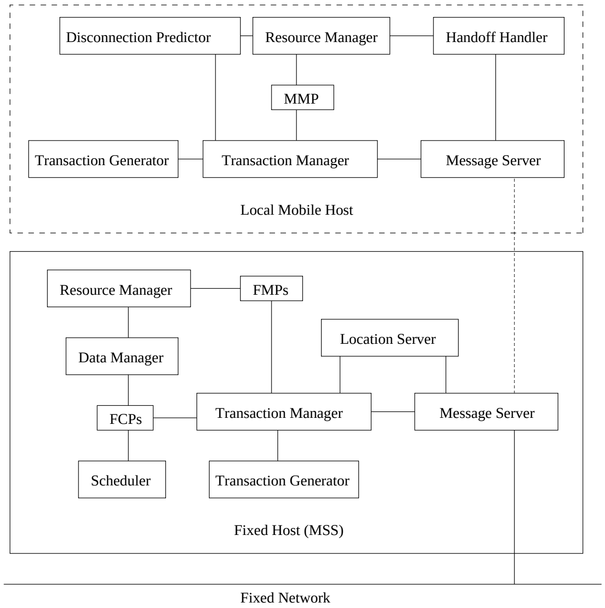

Figure 6: System Components

The components of our simulation model are shown in Figure 6. Each mobile host in our system has a transaction generator, a transaction manager, a message server, a handoff handler, a disconnection predictor, and an MMP for the mobile transaction executed. Each fixed host has a transaction generator, a transaction manager, a message server, a location server, a data manager, a scheduler, an FMP for each transaction it coordinates, and an FCP for each transaction that has submitted an operation to it.

Transactions are generated by the transaction generator according to the transaction modeling parameters and are submitted to the local transaction manager. Deadline and priority assignment of a transaction is also performed by the transaction generator.

The transaction manager at a mobile host initiates an MMP at that host and sends a message to the relevant MSS for the initiation of FMP at the coordinator site. The transaction manager at a fixed host that receives a transaction initiation message first checks whether it is the coordinator site of the generating host of the transaction. If so, it initiates an FMP for the transaction. Otherwise, it sends a request to location server to locate the coordinator site.

Message server is responsible for the transmission of messages between hosts. Sending messages from a site is organized on the basis of the priorities of messages. Mobility management messages have the highest priority in the system. Each transaction operation message carries the priority of its source transaction.

Due to mobility, mobile hosts can change their location and access the fixed network from different points at different times. In order to locate a mobile host and its coordinator from any fixed host, a linked list structure is constructed (Figure 5). The location server is responsible for constructing this list and by using the list it redirects messages between the coordinator site and the current MSS of a mobile host.

Disconnection and reconnection of a mobile host are initiated by the disconnection predictor, and the handoff operation is initiated by the handoff handler by using the relevant parameters of our model.

The execution of transaction operations is controlled by MMP and FMP. FMP provides the execution of *Read* and *Write* operations via FCPs. In order to access a data page, a cohort process first requests a lock on that page from the scheduler. The scheduler is responsible for the serializable execution of transactions by making use of a concurrency control protocol. Our model employs the Priority Abort protocol for concurrency control, as we mentioned before. After a cohort process is granted the lock on a data page, the cohort process is allowed to access the page. If the page is not in main memory, the data manager fetches it from the disk.

I/O and CPU service requests are ordered by the resource manager at each site. The CPU scheduling policy is preemptive-resume priority scheduling, and the I/O scheduling policy is non-preemptive priority scheduling.

5 Experiments

Default parameter settings for the simulation experiments are presented in Table 2. These values were chosen to have a system with high resource utilization at all fixed hosts. Each experiment run continued until 10000 transactions were executed in the system.

ExecStrategy	ESMH, ESFH
NumFHosts	10
NumMHosts	50
ThinkTime	0 second
LocalDBSize	200 pages
PageSize	4096 bytes
MemSize	100 pages
NumFhCPU	2
PageCPUTime	8 mseconds
MsgCPUTime	2 mseconds
CPURatio	2
NumAccessed	8-16 pages
NumUserInt	0
DiskTime	12 mseconds
UpdTrProb	0.5
WriteProb	0.5
SlackRate	5.0
WiredBand	10 Mbps
WirelessBand	2 Mbps
ContMsgSize	256 bytes
HandoffProb	0
DisconProb	0

Table 2: Default Parameter Settings

The performance metric used for the evaluations is *success ratio*; i.e., the fraction of transactions that satisfy their deadlines. In order to be able to interpret performance results, we also kept track of the number of conflicting data access requests per transaction and the average number of mobile host/coordinator site search requests per transaction.

5.1 Transaction Load

In this experiment, we varied the number of mobile hosts (*NumMHosts*) from 10 to 90 in

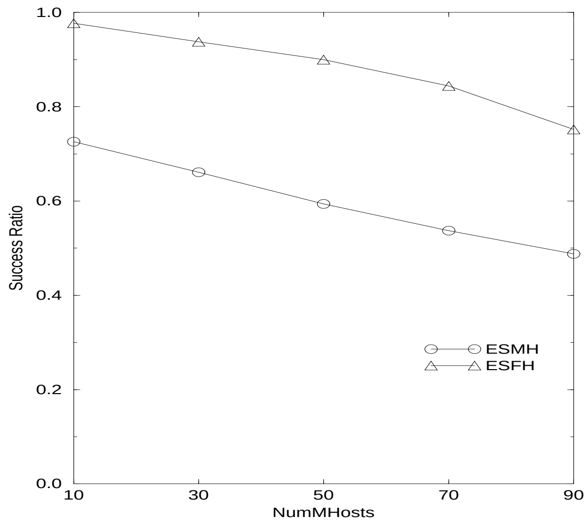

Figure 7: *success ratio* vs *NumMHosts*

increments of 20. Increasing the number of mobile hosts corresponds to an increase in the total system load. The range of transaction load obtained in the experiment led to a CPU utilization of 0.13 to 0.72 for the case of ESMH, and 0.34 to 0.87 for the case of ESFH. The ranges for I/O utilization observed with ESMH and ESFH were 0.17 to 0.84 and 0.36 to 0.87, respectively.

The performance results obtained for the execution strategies ESMH and ESFH in terms of the *success ratio* are shown in Figure 7. Remember that ESMH represents the case where the execution site of the mobile transactions is the mobile hosts, and ESFH represents the case where the execution site is the fixed hosts. As it can be seen from the figure, the performance of the system degrades for both strategies ESMH and ESFH as the transaction load increases, and for all ranges of the transaction load ESFH performs better than ESMH. This is because ESMH involves an additional wireless link delay for each data access request, and uses less powerful CPU of the mobile host.

The reasons for the decrease in the performance with both execution strategies are the increasing load on the physical resources and the increasing number of data conflicts.

5.2 User Interaction

A *user interaction* operation is issued by a mobile host during the execution of a transaction generated by itself. This operation can be considered as reading a local data item that is required to continue the execution of the mobile transaction. Therefore, with ESMH a user interaction request is satisfied locally, and there is no need for wireless link communication. But, with ESFH, a wireless link delay is experienced between the generating mobile host and the current MSS for satisfying the user interaction requirement of the mobile transaction.

In this experiment, we varied the number of user interaction operations for each mobile transaction from 0 to 10 in increments of 2 to evaluate the performance impact of the frequency of mobile user interactions. Performance results obtained in terms of the *success ratio* of mobile transactions are provided in Figure 8.

As the degree of user interactions is increased, the performance of ESMH gets a little bit worse due to the execution overhead of user interaction operation. With ESFH, more

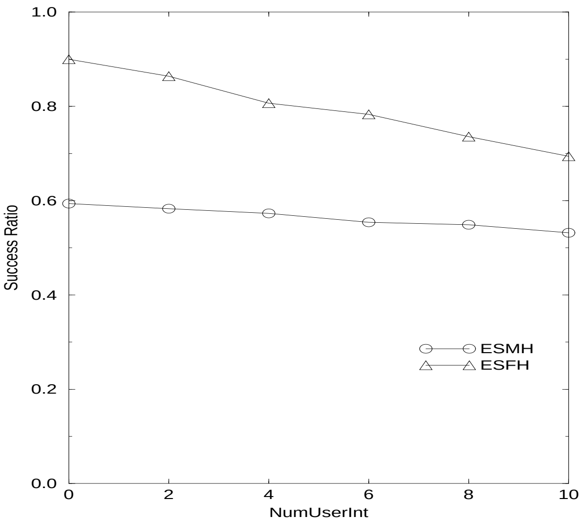

Figure 8: *success ratio* of mobile transactions vs *NumUserInt*

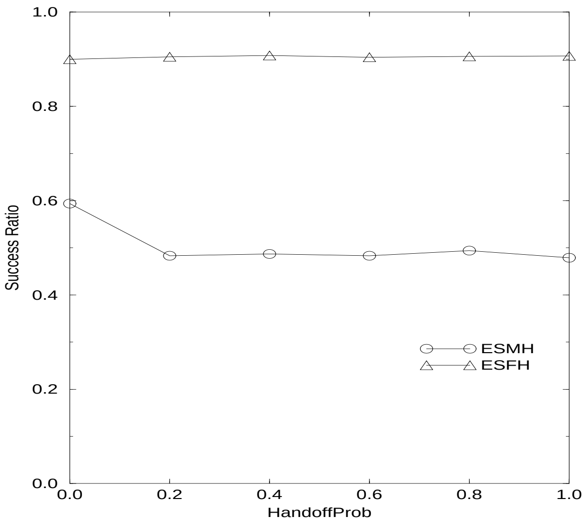

Figure 9: *success ratio* of mobile transactions vs *HandoffProb*

steep degradation in the performance is observed as the number of user interactions increases. This result can be explained by the increasing amount of wireless link delay with each additional interaction request during the execution of mobile transactions.

5.3 Handoff

In a mobile environment, users may cross the boundary between two cells. In order to keep connectivity of a mobile host to the fixed network, a handoff process needs to occur. During the handoff operation, a distributed linked list structure is constructed for the location information of the mobile host that needs to be exchanged between the coordinator site and the current MSS of the mobile host (Figure 5). Therefore, the handoff process introduces a communication and search overhead to the system.

The performance results obtained by varying the handoff probability are shown in Figure 9. In this experiment, simulation time is divided into *HandoffInt* time intervals which has the value of 3 seconds. A handoff process occurs at the beginning of each time interval with probability *HandoffProb*. We varied the handoff probability from 0 to 1 in increments of 0.2. The value of 0 corresponds to the case where no handoff process occurs. In this experiment, we set the value of the number of user interactions to 0.

As we can see, the performance with ESFH is not affected by the frequency of the handoff process. This is because, after the submission of a mobile transaction to its coordinator site, there is no need for message transmission between the coordinator site and the mobile host until the transaction completes. Therefore, the search process for the coordinator site is required too rarely to affect the performance. However, with ESMH, the no handoff case (with *HandoffProb* $= 0$) provides better performance than that with the handoff case (with *HandoffProb* > 0), and for all nonzero values of the handoff probability the performance results are about the same. Remember that we have used the forwarding pointers mechanism for determining the current address of a mobile computer, and as mobile hosts move randomly, after a while the average length of links from the coordinator site to the current MSS becomes the same for all nonzero values of the handoff probability. This means that, the average cost of communication and search between the coordinator site and the current MSS of the mobile host is about the same with any positive value employed for *HandoffProb*. In conclusion, we can say that, although the handoff process leads to a decrease in the performance with ESMH, increasing the frequency of handoffs does not introduce additional degradation in performance.

In order to see how the frequency of user interactions affects the system performance with the handoff process, we varied the number of user interactions from 0 to 10 in increments of 2. In this experiment, *HandoffInt* was assigned the value of 3 seconds and *HandoffProb* was 0.2.

As illustrated in Figure 10, the performance pattern with ESMH is not affected by the change in the number of user interactions. This observation is similar to the results we presented (without handoff) in the preceding section. However, with ESFH, as the number of user interactions increases, the performance degradation of the system becomes sharper compared to that obtained without handoff. This result is due to the increasing cost of the search for the coordinator site. When the number of user interactions is 0, the average number of coordinator site searches per transaction and the average number of mobile host search requests per transaction have the values of 2.12 and 2.23, respectively. But, these two performance measures reach the values of 10.30 and 28.05, respectively, when the number of user interactions becomes 10.

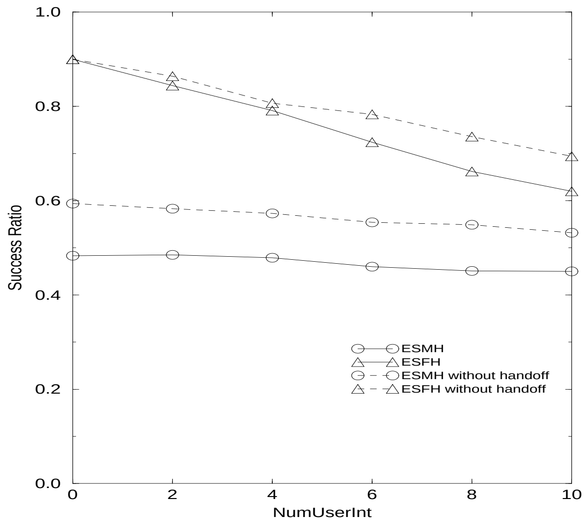

Figure 10: *success ratio* of mobile transactions vs *NumUserInt*

5.4 Disconnection

A mobile host disconnects itself frequently from the fixed part of the system either to save its

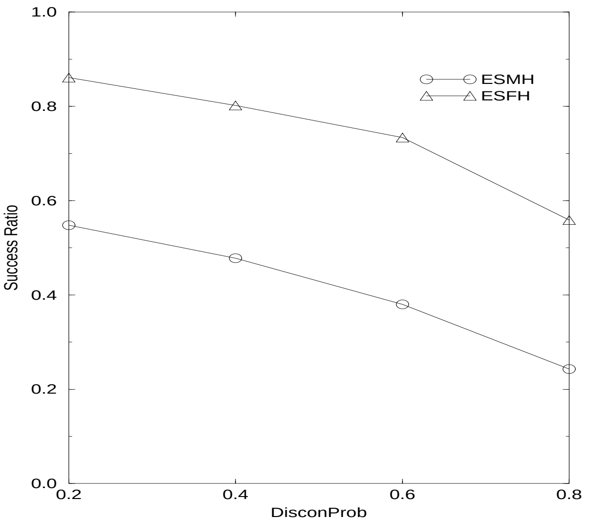

Figure 11: *success ratio* of mobile transactions vs *DisconProb*

limited energy or because of the unavailability of wireless links at some parts. When a mobile host disconnects, it can no longer submit transactions or operations to the system. Therefore, disconnection has a negative impact on the performance of the system.

In this experiment, we evaluated the performance of the system by varying the disconnection probability of the mobile host (i.e., *DisconProb*) from 0.2 to 0.8 in increments of 0.2. Recall that *DisconProb* is the probability that a mobile host is disconnected during the time interval *ConnectInt*. In this experiment, *ConnectInt* was assigned the value of 3 seconds.

As expected, the performance of the system in terms of *success ratio* for mobile transactions degrades while the disconnection probability is increased (Figure 11). However, the relative performance of ESMH and ESFH is not affected by the frequency of disconnections.

5.5 Coordinator Relocation

Due to mobility, the distance between the current MSS of a mobile host and its coordinator site may increase and thus it might be better to relocate the coordinator site to reduce the overhead of communication and search between the current MSS and the coordinator site.

In the experiments of this section, we evaluated the performance impact of relocating the coordinator site of a mobile host. The overhead introduced by the relocation of the coordinator site is due to the exchange of communication messages between the relevant fixed hosts.

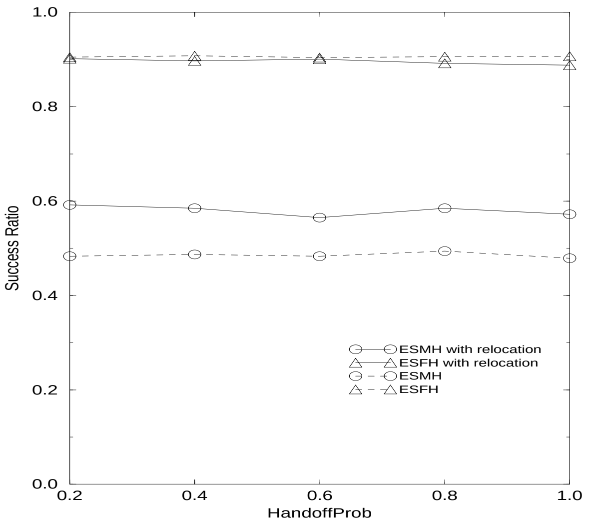

Figure 12: *success ratio* of mobile transactions vs *HandoffProb*

In the first experiment of this section, we measured the performance of the system by varying the handoff probability from 0.2 to 1.0. As we can see in Figure 12, when the execution strategy ESFH is employed, relocating the coordinator site leads to a little bit worse performance. This degradation is due to cost of the relocation. As we explained with the handoff experiment (see Figure 9), the performance of ESFH is not affected by the handoff probability. With ESMH, the performance of the system improves considerably when coordinator relocation is applied, and this improvement is nearly the same for all values of the handoff probability. As the mobile host needs to communicate with the coordinator site for each transaction operation, preventing the cost of search for the coordinator and thus reducing the amount of communication can provide a substantial benefit for the system performance.

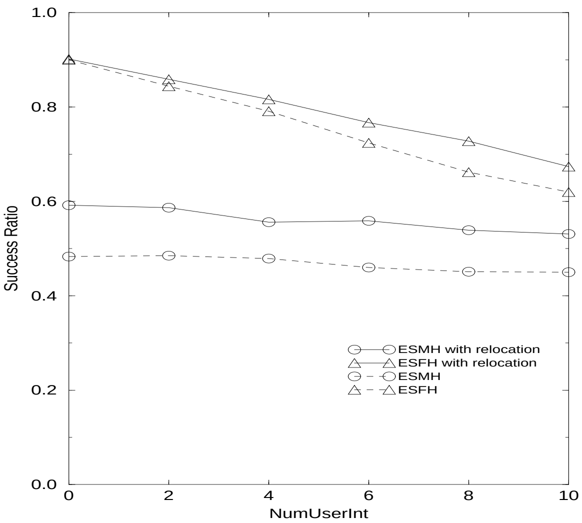

Figure 13: *success ratio* of mobile transactions vs *NumUserInt*

In the second experiment, we varied the number of user interactions from 0 to 10. The performance results are shown in Figure 13. With the execution strategy ESMH, for all values of the number of user interactions, ESMH with relocation performs better than that with no relocation. With ESFH, at lower values of the number of user interactions, relocation does not help the performance of the system improve due to the cost of relocation. At higher values, the cost of communication and search for the coordinator outweighs the cost of relocation, therefore better performance is obtained with relocation.

5.6 Wireless Link Failure

In the experiment of this section, we measured the performance of the system by varying the failure probability of the wireless link. The simulation time was divided into *FailureInt* time intervals and during each interval the wireless connection of a mobile host to the fixed part of the network is failed with probability *FailureProb*.

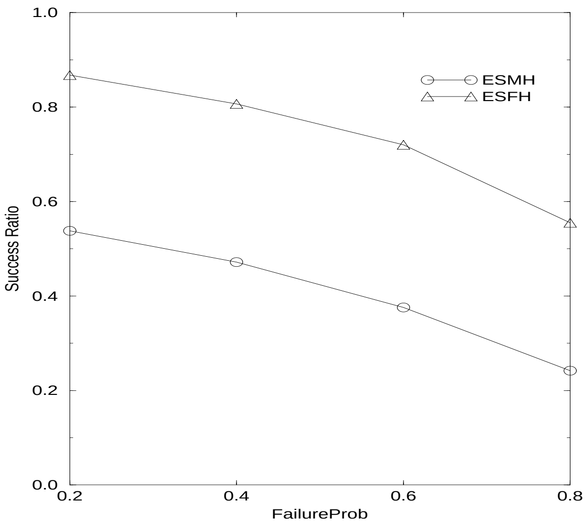

Figure 14: *success ratio* of mobile transactions vs *FailureProb*

The performance results for the experiment are shown in Figure 14. In this experiment, *FailureInt* was set to 3 seconds and the value of failure probability was changed from 0.2 to 0.8 in increments of 0.2. As we can see, as the value of failure probability increases, the performance of the system becomes worse with both execution strategies ESMH and ESFH. The relative performance of ESMH and ESFH is not affected by the probability of wireless link failure. These results are similar to those we obtained with the disconnection experiment (Section 5.4).

6 Conclusions

In this paper, we have provided a mobile database system model that supports real time response to the transactions of underlying applications. We have constructed a mobile transaction execution model and implemented it on a simulation program in order to evaluate the system performance with two different execution strategies, namely, ESMH and ESFH. With the execution strategy ESMH, a data request message for each operation of a mobile transaction is sent to the coordinator site at the fixed part of the network. Upon completion of processing the data access request at the fixed network, the data is provided to the mobile host that has generated the transaction. The transaction operation on the granted data is executed at the mobile host. With the execution strategy ESFH, a mobile transaction is submitted to the coordinator as a whole. The transaction is executed at the fixed host, however, user interaction requests of the transaction are handled by the generating mobile host. In this strategy, the processing requirement of the transaction is satisfied at the fixed part of the network.

We have also performed some experiments to evaluate the impact of the mobile system issues on the performance of the system. Among investigated issues are the handoff process, disconnection, coordinator site relocation, and wireless link failure. The performance metric used in all the experiments was the *success ratio*, i.e., the fraction of transactions that meet their deadlines. It was observed in the experiments that low-power CPU of the mobile hosts and low-bandwidth wireless links are the bottlenecks of the system.

When the performance impact of the handoff process was evaluated, it was observed that, with no user interaction, the performance of the strategy ESFH is not affected by the handoff process. Another observation was that, although the handoff process leads to a decrease in the performance with ESMH, increasing the frequency of handoffs does not introduce additional degradation in performance.

The performance impact of the number of user interactions required for the execution of a mobile transaction was also evaluated. It was seen that with the execution strategy ESFH, increasing the degree of user interaction leads to a degradation in the system performance due to the requirement for wireless link communication for each user interaction.

In another experiment, we investigated the performance impact of relocating the coordinator site of a mobile host. The relocation was aimed to reduce the communication cost between the coordinator and the current MSS of a mobile host. In that experiment, we observed that with ESMH, relocation always leads to an improvement in the performance for any degree of user interaction. With ESFH, as the number of user interactions increases, the performance improvement provided by relocating the coordinator site becomes higher.

And finally, it was observed that the performance impact of disconnections and failures does not show much difference.

Our current work in mobile database systems area includes development of real-time concurrency control and recovery protocols for mobile computing environments, and evaluation of those protocols using the mobile database management system model that we have presented in this paper.

ACKNOWLEDGEMENTS

This research is supported by the Research Council of Turkey (TÜBİTAK) under grant number EEEAG-246 and the NATO Collaborative Research Grant CRG 960648.

References

[1] Abbott, R. K., Garcia-Molina, H. (1992) Scheduling Real-Time Transactions: A Performance Evaluation. *ACM Transactions on Database Systems*, **17**, no.3, 513–560.

[2] Bernstein, P. A., Hadzilacos, V., Goodman, N. (1987) *Concurrency Control and Recovery in Database Systems.* Addison-Wesley, Reading, MA.

[3] Chrysanthis, P. K. (1993) Transaction Processing in Mobile Computing Environment. *Proceedings of the IEEE Workshop on Advances in Parallel and Distributed Systems.*

[4] Cáceres, R., Padmanabhan, V. N. (1996) Fast and Scalable Handoffs for Wireless Internetworks. *Proceedings of ACM MobiCom'96.*

[5] Dunham, M. H., Helal, A. S. (1995) Mobile Computing and Databases: Anything New?. *SIGMOD Record*, **24**, , no.4, 5–9.

[6] Forman, G. H., Zahorjan, J. (1994) The Challenges of Mobile Computing. *IEEE Computer*, **27**, no.4.

[7] Imielinski, T., Badrinath, B. R. (1993) Data Management for Mobile Computing. *SIGMOD Record*, **22**, no.1.

[8] Kayan, E. (1998) *An Evaluation of Transaction Management Issues in Mobile Real-Time Database Systems.* M.S. Thesis, Bilkent University, Turkey.

[9] Őzsoyoğlu, G., Snodgrass, R. T. (1995) Temporal and Real-Time Databases: A

Survey. *IEEE Transactions on Knowledge and Data Engineering*, **7**, no.4, 513–532.

[10] Pitoura, E., Bhargava, B. (1995) Maintaining Consistency of Data in Mobile Distributed Environments. *Proceedings of the 15th International Conference on Distributed Computing Systems.*

[11] Pitoura, E., Bhargava, B. (1994) Revising Transaction Concepts for Mobile Computing *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications.*

[12] Ramamritham, K. (1993) Real-Time Databases. *Distributed and Parallel Databases*, **1**, no.2.

[13] Schwetman, H. (1990) *CSIM User's Guide*. MCC Technical Report Number ACT-126-90.

[14] Ulusoy, Ö. (1994) Processing Real-Time Transactions in a Replicated Database System. *International Journal of Distributed and Parallel Databases*, **2**, no.4, 405–436.

[15] Ulusoy, Ö. (1998) Real-Time Data Management for Mobile Computing. *Proceedings of the International Workshop on Issues and Applications of Database Technology*, 233–240.

[16] Walborn, G. D., Chrysanthis, P. K. (1995) Supporting Semantics-Based Transaction Processing in Mobile Database Applications. *Proceedings of the 14th IEEE Symposium on Reliable Distributed Systems.*

[17] Yeo, L. H., Zaslavsky, A. (1994) Submission of Transactions from Mobile Workstations in a Cooperative Multidatabase Processing Environment. *Proceedings of the 14th International Conference on Distributed Computing Systems.*