AN EVALUATION OF TRANSACTION MANAGEMENT ISSUES IN MOBILE REAL-TIME DATABASE SYSTEMS

A THESIS
SUBMITTED TO THE DEPARTMENT OF COMPUTER
ENGINEERING AND INFORMATION SCIENCE
AND THE INSTITUTE OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By
Ercan Kayan
January, 1998

AN EVALUATION OF TRANSACTION MANAGEMENT ISSUES IN MOBILE REAL-TIME DATABASE SYSTEMS

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER

ENGINEERING AND INFORMATION SCIENCE

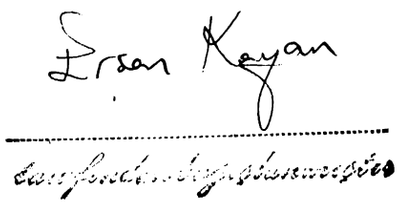
AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE



By

Ersan Kayan

January, 1998

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Özgür Ulusoy (Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Erol Arkun

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Vis. Prof. Dr. Abdullah Uz Tansel

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet Baray, Director of Institute of Engineering and Science

ABSTRACT

AN EVALUATION OF
TRANSACTION MANAGEMENT ISSUES
IN
MOBILE REAL-TIME DATABASE SYSTEMS

Ersan Kayan
M.S. in Computer Engineering and Information Science
Supervisor: Asst. Prof. Dr. Özgür Ulusoy
January, 1998

The integration of issues from real-time database systems and mobile computing systems is a new research area that aims to provide support for real-time requirements of database applications in a mobile computing environment. Due to certain constraints of a mobile computing environment, such as resource-poor mobile computers, low-bandwidth and unreliable wireless links, and mobility of users/computers, various research issues in traditional real-time database systems need to be reconsidered. In this thesis, we investigate some of these issues using a detailed simulation model of a mobile real-time database management system. We propose a transaction execution model with two alternative execution strategies and evaluate the performance of the system considering various mobile system characteristics. Performance results are provided in terms of the fraction of transactions that satisfy their deadlines.

Key words: Mobile Computing, Real-Time Databases, Transaction Execution Model, Performance Analysis.

ÖZET

MOBİL GERÇEK ZAMANLI
BİR VERİTABANI SİSTEMİNDE
HAREKET PERFORMANSININ İNCELEMESİ

Ersan Kayan
Bilgisayar ve Enformatik Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Yrd. Doç. Dr. Özgür Ulusoy
Ocak, 1998

Gerçek zamanlı veritabanı sistemleri ve mobil işletim sistemleri kapsamındaki araştırma konularının entegrasyonu yeni bir araştırma dalı olup, veritabanı uygulamalarının gerçek zamanlı ihtiyaçlarını karşılama amacını gütmektedir. Mobil işletim sistemlerinin belirli bazı sınırlamaları; kaynakça yetersiz mobil bilgisayarlar, düşük kapasiteli ve hata oranı yüksek telsiz hatlar, kullanıcıların ve/veya bilgisayarların hareketliliği gibi, bilinen gerçek zamanlı veritabanı sistemlerindeki birçok araştırma konularının yeniden gözden geçirilmesini gerektirmektedir. Bu tezde, mobil gerçek zamanlı bir veritabanı sisteminin detaylı bir simulasyon programı kullanılarak bu konuların bazıları incelenmektedir. Sistemin performansı önerilen iki değişik işleme stratejisine sahip bir hareket işleme modeli kullanılarak ve değişik mobil sistem özellikleri dikkate alınarak hesaplanmaktadır. Performans sonuçları zaman sınırlamalarının karşılanabilme oranı baz alınarak verilmektedir.

Anahtar kelimeler: Mobil İşletim, Gerçek Zamanlı Veritabanları, Hareket İşleme Modeli, Performans Analizi.

To my family and Rabia

ACKNOWLEDGMENTS

I am very grateful to my supervisor, Asst. Prof. Dr. Özgür Ulusoy for his invaluable guidance and motivating support during this study. His instruction will be the closest and most important reference in my future research.

Finally, I would like to thank the committee members Prof. Dr. Erol Arkun and Vis. Prof. Dr. Abdullah Uz Tansel for their valuable comments.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recent developments in wireless communication technology and portable computers have resulted in the emergence of *mobile computing systems*. In a mobile computing system, users carrying portable (mobile) computers can access database services from any location. Users are not required to maintain a fixed position in the network. The links between mobile computers and the mobile support stations that provide wireless interface to mobile computers can change dynamically. Therefore, a mobile computing system can be viewed as a dynamic type of traditional distributed systems[10]. Local yellow pages, banking, travel information, electronic magazines, electronic mail services, inventory orders are the examples of applications that mobile computing systems can support.

Mobile computers are usually small in size and poor in resource capabilities relative to desktop computers. Many of them completely rely on mobile support stations having scarce main-memory and no disk. On the other hand, there are also some mobile computers that hold databases, and have considerable processing power to support database operations.

Mobile computers use batteries as energy supply. Batteries are required to be replaced or recharged frequently which is rather inconvenient for users. Network connectivity of a mobile computer is provided via a wireless link. Wireless links vary too much in bandwidth, and they are unreliable and costly.

Another constraint for a mobile computer is that it is more prone to failures than a fixed desktop computer.

The constraints that we have mentioned require the system designers to reconsider the issues in traditional distributed systems, such as transaction processing, caching, replication, etc. The limited battery capacity requires energy efficient solutions to the issues. A mobile computer is disconnected frequently in order to save battery energy and reduce communication costs. Disconnections are different from failures, because a disconnection can be anticipated and necessary actions can be taken beforehand. Mobility of computers also necessitates the location management of those computers. In order to contact with a mobile computer, it is required to find out its current location.

Supporting real-time response to underlying applications is another issue that needs to be considered in mobile computing systems as it is pointed out in [23]. A real-time database system supports applications that have timing constraints in terms of deadlines or periods. Satisfying the timing constraints of such applications is the primary goal of real-time database systems.

In this thesis, we present a mobile real-time database system model that takes into account the constraints of mobile computing systems. A transaction execution model we have 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, and so on.

The organization of the thesis is as follows. In Chapter 2, we provide some background information related to mobile computing systems and real-time database management systems. In Chapter 3, our mobile real-time database system model is described in detail. In Chapter 4, the simulation model and the results of performance experiments are presented. Conclusions are provided in the last chapter.

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

Chapter 2

Background

2.1 Mobile Computing Systems

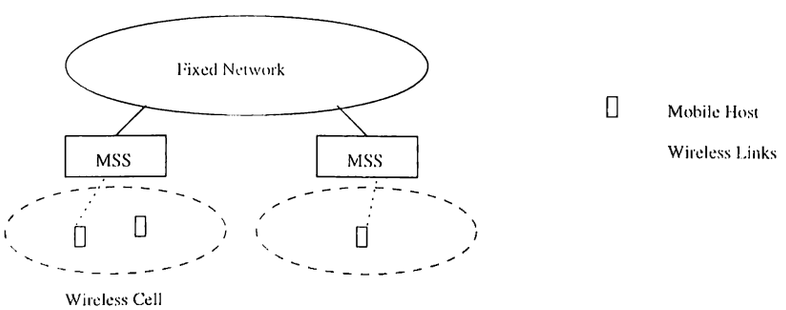


Figure 2.1: A Mobile Computing System

A typical mobile computing system consists of a number of mobile and fixed hosts (Figure 2.1). *Fixed hosts* are connected with each other via a fixed high-speed wired network and constitute the fixed part of the system. A *mobile host* 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 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 data between the local

mobile hosts and the fixed network.

Due to mobility of users mobile hosts 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 handoff the new cell's MSS takes the responsibility of the mobile host from the old cell's MSS. This process should be transparent to the user.

In traditional distributed systems it is assumed that the location of hosts and the connection between them do not change. In mobile computing systems, mobility of hosts invalidates this assumption resulting in need for redesigning algorithms developed for traditional distributed systems[4].

The location of a mobile host can be regarded as a data item that changes as the user crosses the boundary between two cells. At the fixed hosts, location servers maintain the location database for location management. Location servers update the location data of a mobile host when it changes its location and answer queries on location data. In order to contact with a mobile host we first need to find out its location. Broadcast, central services, home bases, and forwarding pointers are the primary mechanisms for determining the current address of a mobile computer[12]:

- *Broadcast*: A search request is broadcasted to all network cells for the mobile computer sought.
- *Central services*: The current address for each mobile computer is maintained in a centralized database.
- *Home bases*: The current location of a mobile host is only maintained at its home location server in which the mobile host is registered.
- *Forwarding pointers*: 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. This method is among the fastest methods.

Mobility of hosts also results in new application types like location dependent queries. The answer to a query on local places, events, etc. can change

according to location of the mobile host that submits the query. An example of a location dependent query is "where is the nearest restaurant?".

Due to mobility, mobile hosts also encounter more heterogeneous network connections.

Mobile hosts operate on batteries which are limited in energy. It is expected that the lifetime of a battery will increase only 20 % over the next 10 years[14]. Hence, energy conservation is an important issue in the design of mobile computing systems.

Network bandwidth is another major performance bottleneck for mobile computing systems. For local area networks bandwidth is in the order of 10 Mbps, and for cellular networks it is in the order of 10 Kbps.

Due to limited battery power and the unavailability of wireless connection mobile hosts can be frequently disconnected. Disconnections can be anticipated and precautions can be taken before a disconnection occurs. Changing signal strength in a wireless network may help to predict a disconnection, and a user may be able to preannounce the power down of the mobile host. Thus, disconnections are different than failures. The Coda File System is an example system that supports disconnected operation in a mobile environment[20]. In Coda, important data are cached before a disconnection, and during disconnection cached data is used. Upon reconnection the cached data are reconciled with the replicated master copy.

In a mobile computing environment mobile hosts are more prone to failures than fixed hosts. For mobile hosts, the connection to the fixed part of the network is primarily provided through a wireless link. Wireless links are relatively unreliable and have low bandwidth constraints.

2.2 Mobile Database System Issues

In a mobile computing environment, the constraints imposed by the mobile computer hardware itself, wireless network technology, and the mobility of users and hosts affect the design of database management systems. Existing protocols, models and algorithms for traditional database systems should be revised in order to adapt to mobile computing environments.

2.2.1 Transaction Processing

A distributed transaction whose operations are executed on mobile and/or fixed hosts of a mobile computing system is called a *mobile transaction*. A mobile transaction is usually initiated by a mobile host and submitted to the system. During execution of the transaction if there is no need for mobile host interaction then the mobile host may disconnect itself from the network in order to save power or reduce the cost of wireless connection. However, disconnection is not always voluntary and it can also be due to unavailability of network connection at some places. Supporting the execution of a mobile transaction while disconnected is an important issue for transaction management in mobile computing systems.

Due to low bandwidth wireless channels and disconnection of mobile hosts, execution of mobile transactions can take a substantial amount of time. Also, due to mobility, the distance between a client and an information provider is not fixed. Therefore, in order to reduce communication cost and improve response time, transactions may be relocated. Mobility can also cause transactions to access heterogeneous database systems.

A possible execution strategy for mobile transactions is discussed in [28]. A transaction is submitted by a mobile host to its mobile support station, it is executed on the fixed part of the system, and then the result is returned to the mobile host. A mobile host may disconnect itself from the network after it submits the transaction to perform some other task. This approach does not support the interactions with the mobile host (e.g., getting input from the

user), and transactions that involve data at mobile host.

A mobile transaction execution model that supports the so called weak operations is proposed in [17, 18]. 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, to maximize local processing in a cluster and reduce network access, the user is allowed to interact with locally available consistent data. Therefore, 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. Disconnected operation is supported by weak operations.

The mobile transaction model proposed in [8] is an open-nested transaction model that supports reporting transactions and co-transactions in order 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 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. Disconnected operation is not supported in this model.

In [26], 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. Disconnected operation 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 and unilateral commitment of transactions on the mobile host.

2.2.2 Query Processing

In a mobile computing environment, parameters to a query may be expressed relative to the current location of a mobile host. Location dependent databases maintain information about local services such as information about motels, restaurants, etc. Location dependent queries that are generated on these databases can return different results depending on the location of a mobile host [10]. The answer to a query may also change depending on the state of a mobile host (i.e., whether it is connected or disconnected).

Location information stored in a database may not be complete. Therefore, in order to answer a location dependent query, additional data acquisition may be required during the run time of the query[13].

Query optimization techniques for mobile computing systems should consider the low bandwidth wireless links and the limited battery power. Approximate answers to queries are more acceptable in mobile environments than that in traditional database systems due to constraints of mobile environments[3].

2.2.3 Disconnection

As we discussed before, disconnection of a mobile host can be known beforehand. Therefore, some special actions can be taken on behalf of active transactions. Among those actions are migrating transaction process to a fixed computer, caching remote data that is necessary for the continuation of transaction execution, and transferring log records to a fixed computer.

2.2.4 Data Broadcasting and Caching

Data Broadcasting

Due to low bandwidth and limited power constraints of mobile computing environments, providing data to massive numbers of mobile users is a new challenge to researchers. There are two possible ways to provide data to the users on the communication channel which consists of a downlink channel from servers to clients and an uplink channel from clients to servers[15]:

- *Data Broadcasting*: Broadcasting data on the downlink channel periodically. Since the cost of data broadcasting does not depend on the number of users, broadcasting the most frequently requested data is an effective way of providing data to mobile users.
- *Interactive/On-Demand*: Data are sent to the client on the downlink channel upon a request of client on the uplink channel.

Data broadcasting techniques proposed in [15] multiplex index information that indicates the point of time in the broadcast channel when particular records are broadcasted with data on the wireless channel. Thus, clients may remain in doze mode most of the time. In order to download the required data, they tune in periodically to the broadcast channel. It is concluded that index based organization and access to the data broadcasted over wireless channels conserves battery energy.

The *broadcast disks* technique is proposed in [2] for non-uniformly accessed data. In this technique, data are partitioned into multiple ranges of similar access probabilities. These ranges are referred to as *disks*. Chunks of data from different disks are multiplexed to create the broadcast in such a way that the chunks of the fast disks, disks that consist of hot pages, are repeated more often than the chunks of the slow disks, disks that consist of cold pages. It is shown by simulation experiments that significant improvement in performance can be gained by the broadcast disk. Also, the effects of the broadcast disk on the caching strategy are discussed.

Caching

Caching of frequently accessed data items in mobile computing environments is important in order to reduce contention on the low bandwidth wireless channels. But, caching in mobile environments affects the cache invalidation strategies due to disconnection and mobility of mobile hosts.

When broadcasting is used to invalidate caches, a server periodically broadcasts an *invalidation report*. An invalidation report includes limited or partial information regarding data changes. Due to disconnection, a mobile host may miss some of the invalidation reports, resulting in *false invalidation* of mobile host cache. In this case, the mobile host may invalidate its cache even if it is valid.

In [5], three new cache invalidation methods for mobile computing environments are proposed. In these strategies, the server periodically broadcasts an invalidation report and the clients listen to these reports. Clients do not check their cache validity with the server. Therefore, if the disconnection of a client has been too long, false invalidation may occur.

The scheme proposed in [27] to invalidate caches checks the cache validity with the server, if necessary. In this scheme, the server partitions the database into a number of groups, and also dynamically identifies hot update set that has been updated in a group. After a reconnection, a mobile host checks its cache validity with the server at a group level to save uplink costs. The server validates a group by excluding the hot update set. It is concluded by simulation experiments that this scheme requires much less bandwidth for processing queries, particularly if a mobile computer is occasionally disconnected for a long period of time and most of the data objects are infrequently updated.

Two adaptive caching algorithms are proposed in [7]. The first one adapts to changing environmental parameters by adjusting the window size of the database according to changes in the false invalidation and update rates. The second one differs from the first one in that data items are dynamically partitioned into two groups, one consisting of hot items of updates and the other of cold items of updates. Different window sizes are assigned to the two groups

dynamically on the basis of feedback regarding the false invalidation ratio of client caches.

2.3 Real-Time Database Management Systems

Applications in a real-time database system are supported by providing timely access to the underlying database. The requirement of timely access of applications comes from the characteristics of the applications and/or the data that they are accessing. For example, telephone switching systems, network management systems, stock market, banking, airline reservation systems, etc. require timely response, while accessing archival data and temporal data which loses its validity after a certain time.

Tasks in a real-time database system possess certain time constraints in terms of periods or deadlines, and the satisfaction of these constraints is the primary goal of the system.

Research issues in real-time database systems are discussed in [16, 19, 24]. Most work done in the field involves the scheduling of transactions that have time constraints. The rest of the studies includes managing I/O and buffers, replication, security, and recovery. In the following sections we concentrate on the transaction scheduling and replication aspects of the real-time database management systems.

2.4 Transaction Scheduling

The objective of scheduling policies in real-time database systems is to minimize the number of transactions that missed their deadlines. For most applications it is also desirable to maintain the consistency of the underlying database.

Scheduling of transactions is much harder than that in the conventional database systems due to the time constraints of the transactions. Because,

timely execution of transactions requires a good estimate of their worst-case execution time which is rather difficult to obtain in database systems due to

- dependence of transaction's execution sequence on data values,
- data and resource conflicts,
- I/O scheduling and buffer management techniques,
- communication delays and failures, and
- rollbacks and restarts after transaction aborts, etc [19, 16].

Factors that characterize transactions in real-time database systems have influence on the scheduling policies. Transactions can be characterized according to the effect of missing their deadlines:

- *Hard deadline transactions*: Missing a deadline may result in a catastrophe for a hard deadline transaction.
- *Soft deadline transactions*: Soft deadline transactions have some value even after their deadlines. Even if a soft deadline transaction misses its deadline it is allowed to complete.
- *Firm deadline transactions*: Firm deadline transactions have no value after their deadlines expire. Thus, a transaction that missed its deadline is aborted and permanently discarded from the system.

Transactions can also be characterized according to their data access types, their arrival patterns, knowledge of data items to be accessed, and knowledge and CPU and I/O time to spend.

2.4.1 Priority Assignment

Each real-time transaction is assigned a priority based on its timing constraint and the value of finishing it by its deadline. Priorities for real-time transactions are used for conflict resolution and scheduling of resources. Some of the policies used for priority assignment in real-time database systems are [19, 1]:

- *first come first served*: The transaction with the earliest arrival time is assigned the highest priority. The primary weakness of this policy is that it does not make use of deadline information.
- *earliest deadline first*: The transaction with the earliest deadline is assigned the highest priority.
- *highest value first*: The transaction that imparts the highest value to the system in case of a successful termination has the highest priority.
- *least slack time first*: The slack time is an estimate of how long we can delay the execution of a transaction and still meet its deadline. This policy assigns the highest priority to the transaction with the least slack time.

Priority assignment policy is important because it affects the performance of transaction scheduling algorithms.

2.4.2 Concurrency Control

In real-time database systems, for most applications it is desirable to maintain database consistency while satisfying the timing constraints of transactions. Serializability ensures the consistency of database by controlling the execution of concurrent transactions [6]. Concurrency control techniques resolve conflicts between transactions that want to access the same data object at the same time.

Concurrency control techniques that use serializability as the correctness criteria in real-time database systems are the time-cognizant extensions of lock-based, optimistic, and timestamp-based protocols that have been proposed for

the conventional database systems.

Lock-based protocols require each transaction to obtain a shared lock on each data item they read, and an exclusive lock on each data item they write. Conflicting requests are resolved by using priorities of conflicting transactions resulting in transaction blocking or transaction aborts. The situation in which a higher-priority transaction is blocked by a lower-priority one during conflict resolution is called *priority inversion*. The basic approaches proposed for avoiding this situation are called the *Priority Inheritance* and the *Priority Abort* [24].

In the Priority Inheritance approach, the lower-priority transaction that is blocking other higher-priority transactions inherits the highest priority of the blocked transactions until it releases the lock. Inherited priority helps to execute transaction faster resulting in reduced blocking times for high priority transactions. But, under high data contention the performance of the system degrades rapidly, because in this situation priority inheritance results in most transactions executing at the same priority.

In the Priority Abort approach conflicts are resolved in favor of higher-priority transactions. If the transaction requesting the lock has higher priority than the transaction holding the lock, the higher-priority transaction is granted the lock after the lock-holding transaction aborts. Otherwise, the lock-requesting transaction waits. This approach is deadlock free if the priority of transactions does not change during their execution and the priorities of the transactions are unique. Also, this approach prevents priority inversion, because a higher-priority transaction never waits for a lower-priority transaction.

It is concluded in [25] that aborting a low priority transaction is preferable in real-time database systems to blocking a high priority transaction by inheriting its priority. On the other hand, Priority Abort approach wastes more resources due to aborting transactions.

Timestamp-based protocols assign timestamps to transactions based on their start time for resolving conflicts. This protocol is not a good choice for real-time database management systems, because the timestamp order has

no relationship to transaction priority order.

With optimistic concurrency control protocols, transactions are validated for commitment after they complete their execution. Backward validation or forward validation is performed by the protocol to validate transactions [16, 19]. In backward validation, the validating transaction is aborted if it has conflicts with transactions that have already committed. Otherwise, it is committed. In forward validation, either the validating transaction or currently active transactions that have conflicts with the validating transaction are aborted.

Although the optimistic concurrency control protocol is nonblocking and deadlock-free, transaction aborts and restarts waste resources.

2.4.3 Commitment of Transactions

In distributed real-time database systems, atomic commitment property ensures that the effects of a distributed transaction on the data result at all sites in all or nothing fashion. In these systems, the simplest and most popular atomic commitment protocol used is the two-phase commit protocol [6].

2.5 Replication

Data replication is important in real-time database systems due to high data availability and improved read performance it provides. However, in a replicated database system, access to a data item is distributed across the sites each has a copy of the item. It is necessary to ensure the mutual consistency of the replicated data, i.e., to ensure that all copies of a replicated data behave like a single copy. Communication overhead experienced while updating the multiple copies of a data item may degrade the performance of the system.

In [22], the impact of replication in a distributed database system on satisfying the timing constraints of real-time transactions is evaluated. In that work, different application types that are characterized by the fraction of update transactions processed and the distribution of accessed data items are

considered. It is shown that, for applications where majority of transactions are read-only replication improves the system performance. But, for update intensive applications, due to the overhead of update synchronization among the multiple copies of the updated data, replication causes the system performance to decrease. Another result obtained is that replication improves performance of the system as site failures become more frequent.

Chapter 3

A Mobile Real-Time Database System Model

In [13], research issues for mobile data management are discussed under the categorization of mobility, disconnection, new data access modes, and scale. In [23], the issue of real-time support for mobile applications is also considered. Providing timely response to transactions of the underlying application is important in mobile computing systems. For example, in mobile environments, the location information of a mobile host can be treated as a dynamically changing value. Therefore, a query on the location information of a mobile host should be associated with a timing constraint to produce precise answers.

The constraints of mobile computing systems make it difficult to meet timing constraints of the applications. Low bandwidth and unreliable wireless links, and frequent disconnections increase the overhead of communication of mobile hosts with 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.

In order to have a reasonable performance for a mobile real-time database system, research issues in conventional real-time database systems should be reconsidered to overcome the constraints imposed by the mobile computing systems.

In this chapter, we propose a real-time database system model suitable for mobile computing environments.

3.1 A Mobile Real-Time Database System Model

In our mobile computing system model, we assume that all fixed hosts can be considered as mobile support stations (MSSs) (See Figure 2.1). All system components are assumed to be failure free[1]. Each MSS has a database server and a database consists of pages. Strict data consistency is enforced in our system.

In our system, 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. However, due to mobility, distance between the current MSS of a mobile host and its coordinator site may increase and therefore it might be better to relocate the coordinator site to reduce the communication and search overhead between the current MSS of the mobile host and the coordinator site. We will also devote a section to the discussion of the performance impact of relocation of the coordinator site in the next chapter.

3.1.1 Transaction Execution

A mobile real-time transaction is a distributed transaction that has a timing constraint in terms of a deadline, and is executed at the generating mobile host and possibly at some fixed hosts. Hereafter, we will use the terms transaction and mobile transaction interchangeably.

Our transaction execution model is an extension of the model in [22] to mobile computing systems. In our model transactions are generated by mobile hosts and submitted to the system. Each mobile host is allowed to submit a

[1]But, as we mentioned before, one of the characteristics of a mobile computing system is that mobile hosts and the wireless links are prone to failures. Therefore, we will also investigate the impact of the wireless link failure in a separate section in the next chapter.

transaction only after the previous transaction has finished. So, each mobile host has at most one transaction at a time in the system.

Each 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 site of the generating mobile host, and cohort processes at sites where data operations are executed. Each transaction can have at most one cohort process at a site.

Transactions consist of *Read*, *Write* and *user interaction* operations. Control and data request messages for each *Read* and *Write* operation 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 sites. 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. The data items are accessed randomly by a transaction. A user interaction operation is handled at the generating host of the transaction. It can be considered as reading a local data item from the mobile host that generates the transaction.

A mobile transaction may be executed in one of two ways:

1. The entire transaction may be submitted in a single request message to the fixed network (Figure 3.1). In this case 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 will call this strategy for executing the transactions *ESFH (Execution Site is a Fixed Host)*.

2. Control or data request message for each *Read* and *Write* operation of a transaction may be submitted one after another to FMP by MMP(Figure 3.2). FMP coordinates each request in the fixed network and acknowledges the result to MMP. Processing of each operation is performed at the mobile host. This execution strategy will be called *ESMH (Execution Site is a Mobile Host)*.

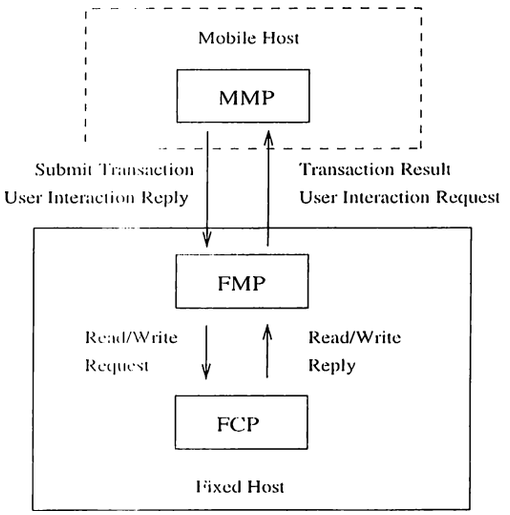


Figure 3.1: Transaction Execution Strategy ESFH

In the first approach, CPU power of the mobile host is not used for processing transaction operations making it more suitable for mobile hosts which do not have powerful CPUs.

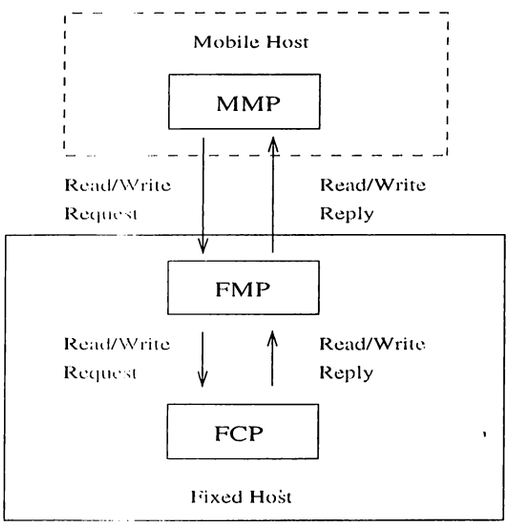


Figure 3.2: Transaction Execution Strategy ESMH

Each 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 be firm deadline transactions (i.e., the transaction that has missed its deadline is aborted and discarded from the system[16, 19]). Both FMP and MMP can initiate the abort when the deadline of the transaction expires. We assume that the system has no pre-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 before its deadline expires because of a data conflict, it is restarted with the same deadline and priority.

3.1.2 Concurrency Control

Serializability is ensured by the two-phase locking protocol. Priority Abort approach is used for resolving data conflicts [1]. Each fixed site in the system has a scheduler to manage the lock requests of the cohort processes. Each cohort process executing at a fixed site has to obtain a shared lock on each data item it reads, and an exclusive lock on each data item it writes. If a lock request is denied, the cohort process requesting the lock is blocked until the lock is released.

If a data item is locked in shared mode by one or more cohort processes, a cohort process requesting a shared lock on the data item is granted the lock if its priority is greater than the maximum priority of all cohort processes, if any, which are waiting to lock the data item in exclusive mode.

Global serializability is provided by holding the locks of a transaction until the transaction has been committed.

3.1.3 Atomic Commitment

Atomic commitment of a transaction is provided by using a modified version of two phase commit (2PC) protocol [6]. In this protocol, FMP is designated as the coordinator, and each cohort process executing on a data site acts as a participant. The protocol can be described as follows:

- If MMP has the control of the transaction execution, it initiates the commit by sending a vote request to FMP.

- If FMP has the control of the transaction execution, it initiates the commit by sending a vote request message to all participants. Otherwise, after receiving the vote request message from MMP, it sends a vote request message to all participants.

- When a participant receives a vote request message, it responds by sending to FMP a message containing that participant's vote : *Yes* or *No*. If the participant votes *No*, it decides *Abort* and stops.

- FMP collects the vote messages from all participants. If all of them were *Yes*, and the FMP's vote is also *Yes* then FMP decides *Commit* and sends commit messages to all participants. Otherwise, FMP decides *Abort* and sends abort messages to all participants that voted *Yes*. In either case FMP sends the decision to MMP and stops.

- Each participant that voted *Yes* waits for commit or abort message from the coordinator. When it receives the message, it makes its decision accordingly and stops.

- When MMP receives commit or abort message from FMP, it makes its decision accordingly and stops.

If FMP does not have the control of transaction execution, it receives a message from MMP for the initiation of commit operation. Due to disconnection of the mobile host, FMP may not be able to receive this message before the deadline of the transaction expires. In such a case FMP can unilaterally abort the transaction in order to prevent data items to be kept locked unnecessarily a long time by the transaction.

A transaction is assumed to be committed when FMP decides commit. After FMP sends the vote request message to all participants, it is not allowed to abort the transaction in case of a data conflict.

When a cohort process commits it writes the updates into the local database. If the cohort process aborts then the updates are just ignored. Before the termination of the cohort process all its locks are released.

3.1.4 Handoff

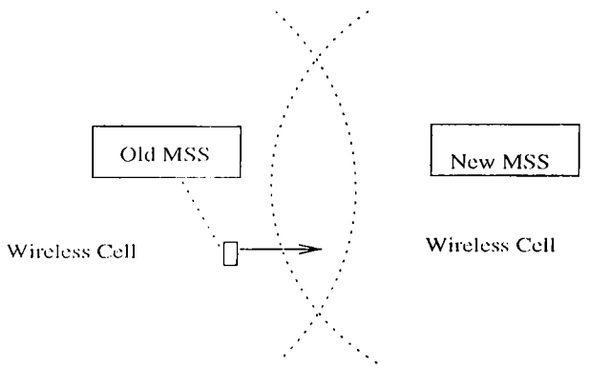


Figure 3.3: Handoff Operation

Due to mobility of users mobile hosts may cross the boundary between two cells (Figure 3.3). In order to keep connectivity of a mobile host to the fixed network a handoff process occurs. During handoff the new cell's MSS takes the responsibility of the mobile host from the old cell's MSS. This process should be transparent to the user.

Our handoff protocol is similar to the one described in [9]. 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 old MSS.

Handoff process goes as follows :

1. The mobile host sends a handoff request message that contains the address of the mobile host and that of the old MSS to the new MSS. If the coordinator site is fixed, it also sends the address of the coordinator site.

2. When the new MSS receives the handoff request of the mobile host, it updates its location table and sends a notify message to the old MSS to inform it.

3. When the old MSS receives the notify message of the new MSS it updates its location table so that the messages for the mobile host can be redirected

to the new MSS. The old MSS acknowledges with the address of the coordinator site of the mobile host if the coordinator site is not fixed.

4. When the new MSS receives the acknowledgment message, it updates its location table to redirect messages from the mobile host to the coordinator site of the mobile host.

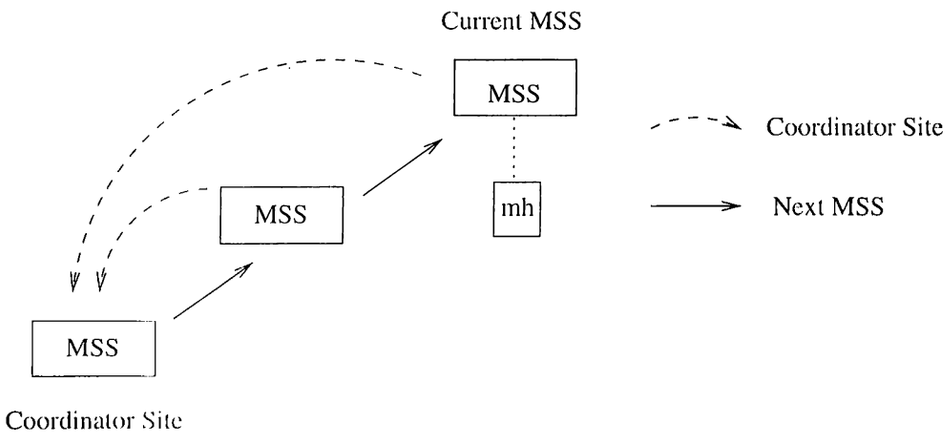


Figure 3.4: Linked List Structure Between Coordinator Site and Current MSS

In this way, as shown in Figure 3.4, 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.

3.1.5 Disconnection

One of the characteristics of mobile computing systems is frequent disconnection of mobile hosts due to limited battery power, unavailability of wireless links, and high monetary cost of wireless links. Supporting disconnected operation is an important issue in mobile database systems. Disconnection of a mobile host can be anticipated and prepared before the disconnection occurs. Our database system model supports disconnected operation only by completing the transmission of the message that is being transmitted on the wireless link. But, during the disconnection period, communication between FMP and MMP is not possible.

In order to prevent a transaction to block other transactions due to a data conflict after its deadline expires, FMP has right to unilaterally abort the transaction even if it does not have the execution control of the transaction. If FMP has the control of the transaction execution it can commit or abort the transaction during disconnection. Upon reconnection of the mobile host it informs MMP about transaction abort or commit.

Chapter 4

Simulation Experiments

4.1 Simulation Model

We have implemented our mobile real-time database system model on a simulation program written in CSIM[21], a process oriented simulation language, in order to evaluate the system performance on different parameter settings.

The simulation model parameters are listed in Table 4.1. 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 fixed part via wireless links over the MSSs.

For each message sent/received by any host *MsgCPUTime* msecs CPU time is used. We assume that there is no contention for wireless link bandwidth. The size of a control message is *ContMsgSize* bytes.

Each fixed host has *NumFhCPU* CPUs and a disk. These resources are shared by all users. Each mobile host has only one CPU and this CPU is used only by the mobile user himself. *PageCPUTime* is the time required to process a page on a fixed host, and *DiskTime* is the time required to access a page on disk. The time required to process a page on a mobile host is *PageCPUTime* * *CPURatio*.

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

Table 4.1: Simulation Parameters

Each fixed host has a local database of size *LocalDBSize* pages. *MemSize* is the memory size of a fixed host in number of data pages.

Each transaction consists of *Read*, *Write* and *user interaction* operations. Number of *Read* and *user interaction* operations are *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 in order to resolve conflicts and schedule resources. The priority assignment policy used in our model is *earliest deadline first*.

Transactions are submitted by each mobile 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*. A mobile host changes its location at the beginning of each *HandoffInt* time interval with probability *HandoffProb*. A mobile host stays connected/disconnected on the average *ConnectInt* seconds, 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*).

4.1.1 Deadline Assignment

Let *ArrivalTime*(T), *ExecTimeEst*(T), and *SlackTime*(T) denote the arrival time, execution time estimate, and slack time of a transaction T, respectively. The deadline of the transaction T is determined, similar to [22], by the formula:

$$Deadline(T) = ArrivalTime(T) + ExecTimeEst(T) + SlackTime(T)$$

The slack time of a transaction is chosen randomly from an exponential distribution with a mean of *SlackRate* times the estimated execution time of the transaction.

In calculating the execution time estimate of a transaction, we assumed an unloaded system where *ExecStrategy* is ESFH. The values used for other parameters in calculation are the ones given in Table 4.2.

4.1.2 System Components

The components of our simulation model are shown in Figure 4.1. Each mobile host in our system has a transaction generator, a transaction manager, a message server, a handoff handler, a disconnection predictor, and for the mobile transaction executed an MMP. Each fixed host has a transaction manager, a message server, a location server, a data manager, a scheduler, and for

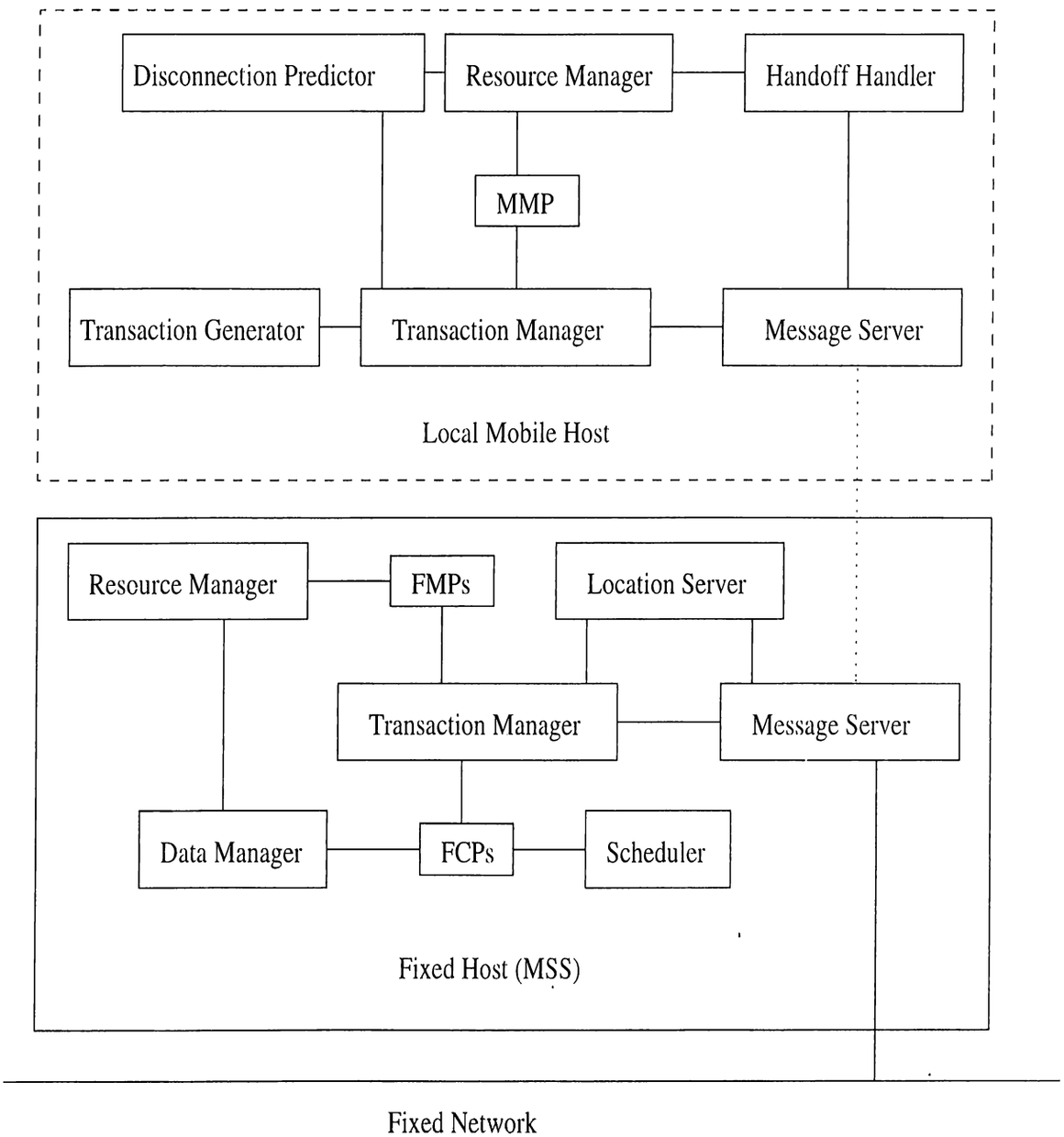


Figure 4.1: System Components

each mobile host it coordinates, an FMP and for each transaction that has submitted an operation to it, a cohort process (FCP).

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

The transaction manager at a mobile host initiates an MMP at the mobile host and sends a message to the MSS of the mobile host for the initiation of FMP at the coordinator site. The transaction manager at the MSS that receives an FMP initiation message first checks whether it is the coordinator site of the mobile host that has sent the message. If so, it initiates an FMP for the transaction. Otherwise, it sends the request to location server to locate the coordinator site. Operation request and reply messages are directed to relevant master and cohort processes.

Message server is responsible with the transmission of messages between hosts. Sending messages from a site is organized based on the priorities of messages. Mobility management messages have the highest priority in the system. Each transaction operation message has the priority of the associated transaction. Wireless network is assumed to have no contention.

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, during handoff operation a linked list structure is constructed (Figure 3.4). The location server is responsible for constructing this list and by using this list it redirects messages between the coordinator site and the current MSS of a mobile host. When the coordinator site relocation is used, location server is also responsible for relocating the coordinator site.

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

Operations of a transaction is controlled by MMP and FMP. FMP provides

the execution of *Read* and *Write* operations via FCPs.

In order to access a page, a cohort process first requests a lock from the scheduler on the relevant data page. Scheduler is responsible for the serializable executions of the transactions by making use of a concurrency control protocol.

After a cohort process is granted the lock on a data page, the cohort process can access that page. If the page is not in memory, then the data manager transfers the page from the disk into memory. It is assumed that a page is in memory with the probability

$$MemSize/LocalDBSize$$

if *MemSize* is less than or equal to *LocalDBSize*. Data manager uses the priorities of the transactions in handling page transfer requests.

I/O and CPU services are ordered by the resource manager at a site. CPU scheduling policy is preemptive-resume priority scheduling, and I/O scheduling policy is non-preemptive priority scheduling.

4.2 Experiments

Default parameter settings for each of the simulation experiments are presented in Table 4.2. These values were chosen to have a system with high resource utilization at all fixed hosts. Access to the following list of resources can lead to contention :

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

Table 4.2: Default Parameter Settings

- CPUs at a fixed host,
- disk of a fixed host,
- wired link, and
- database.

In our system, we assume no contention on the CPU of the mobile hosts and the wireless links.

Each experiment run until 10000 transactions are executed in the whole system. We have performed experiments for both execution strategies ESMH and ESFH.

The performance metric used for the evaluations is the *success ratio*, i.e., the ratio of the number of transactions committed to the total number of transactions generated. In order to be able to interpret performance results, we have also calculated the following performance metrics for each experiment:

- *Restart Ratio*: Average number of restarts experienced by each transaction.
- *Data Conflict Ratio*: The ratio of the number of conflicting data access requests to the total number of data access requests by all transactions.
- *Coordinator Site Search Ratio*: Average number of coordinator site search requests per transaction.
- *Mobile Host Search Ratio*: Average number of mobile host search requests per transaction.

We have conducted experiments in two separate parts. The first part consists of the experiments that evaluate the performance impact of the system parameters, and the second part includes an investigation of various mobile system issues, namely, handoff, disconnection, relocation of transaction coordinator, and wireless link failure.

4.2.1 Impact of System Parameters

Transaction Load

At any time instant, each mobile host can have at most one transaction executing in the system. Therefore, increasing the number of mobile hosts corresponds to an increase in the system load. The maximum number of transactions in the system is *NumMHosts* at any time.

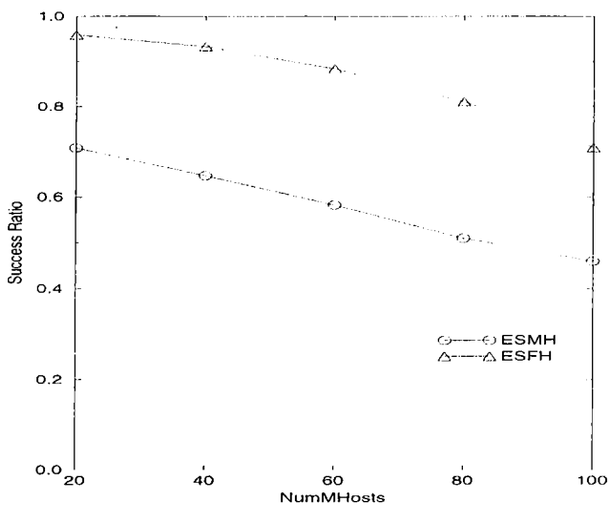


Figure 4.2: Success Ratio vs NumMHosts

In this experiment, we varied the number of mobile hosts (*NumMHosts*) from 20 to 100 in increments of 20. This range of transaction load leads to a CPU utilization of 0.28 to 0.76 for the case of ESMH, and of 0.59 to 0.89 for the case of ESFH. The ranges for I/O utilization observed with ESMH and ESFH are 0.35 to 0.89 and 0.62 to 0.89, respectively. We have chosen the upper value for the transaction load as 100 because for the larger values of transaction load, the system becomes overloaded and unstable. The other parameters were assigned the default values specified in Table 4.2.

Under these parameter settings the performance of the system for the strategies ESMH and ESFH in terms of the success ratio is shown in Figure 4.2. Remember that ESMH represents the case where the execution site is the mobile host, and ESFH represents the case where the execution site is the fixed host. As it is seen, the performance of the system degrades for both the 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.

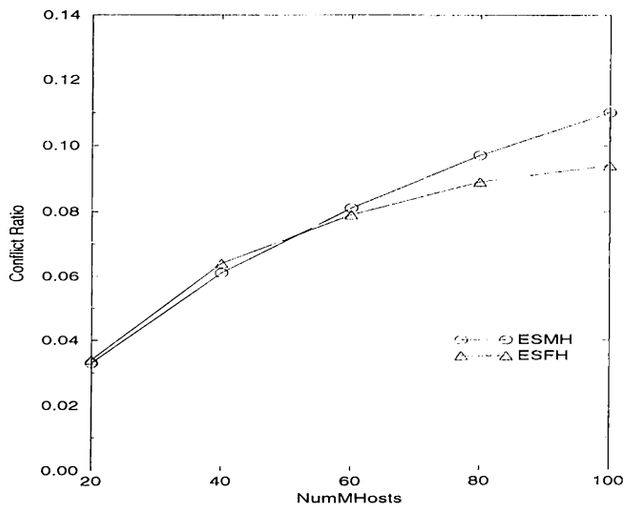


Figure 4.3: Conflict Ratio vs NumMHosts

The reasons for the decrease in the performance of both strategies ESMH and ESFH are the increasing load on the physical resources and the increasing number of conflicts. Figure 4.3 shows how the conflict ratio changes in these experiments. For both ESMH and ESFH, as the system load increases, the conflict ratio also increases due to the increasing number of data access requests, and hence increasing probability of conflict between the concurrent transactions. At lower levels of the system load, both ESMH and ESFH experience about the same number of conflicts. But at higher system loads, the number of conflicts with ESMH becomes higher and the difference between the results obtained with ESMH and ESFH also increases. Because, in case of ESFH, at higher loads, the number of transactions whose deadlines expire due to high CPU contention increases, resulting in reduced number of data access requests by the active transactions.

The effect of increasing CPU contention can also be seen by looking at the restart ratios (Figure 4.4). Although at lower levels of transaction load restart ratio of ESFH is higher than that of ESMH, at higher levels of load the situation is the opposite. The increasing ratio of the restarts at lower levels for

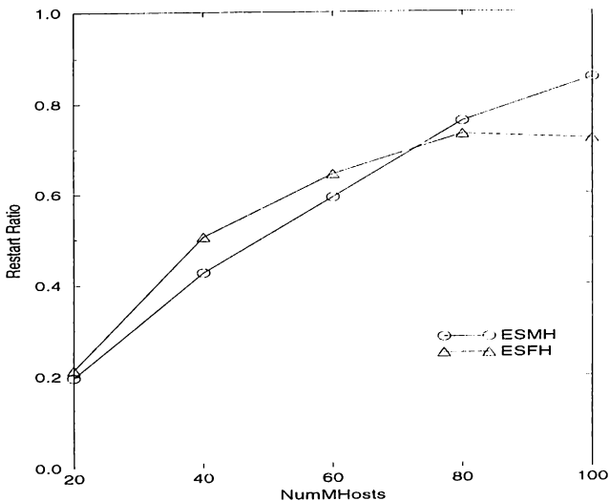


Figure 4.4: Restart Ratio vs NumMHosts

both ESMH and ESFH is due to the increasing data conflicts. But at higher levels of the load, high contention of the CPU in case of ESFH overweights the effect of conflicts and some of the transactions aborted due to a data conflict can not be restarted. Therefore, the number of restarts does not increase at higher loads for ESFH.

Think Time

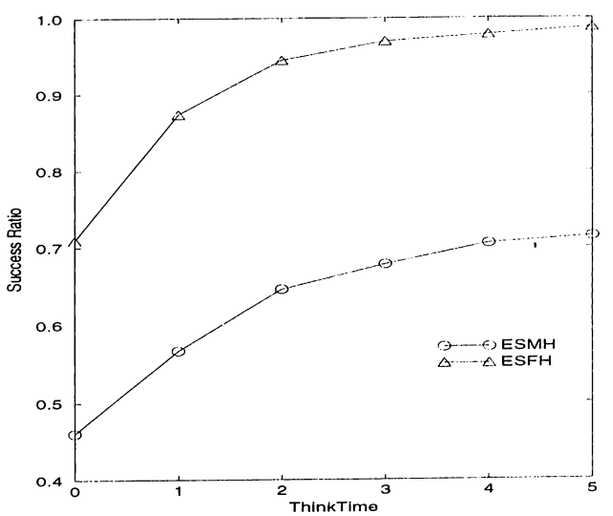


Figure 4.5: Success Ratio vs ThinkTime

Increasing the think time reduces the number of active transactions in the system at any instant. This results in less data and resource contention in the

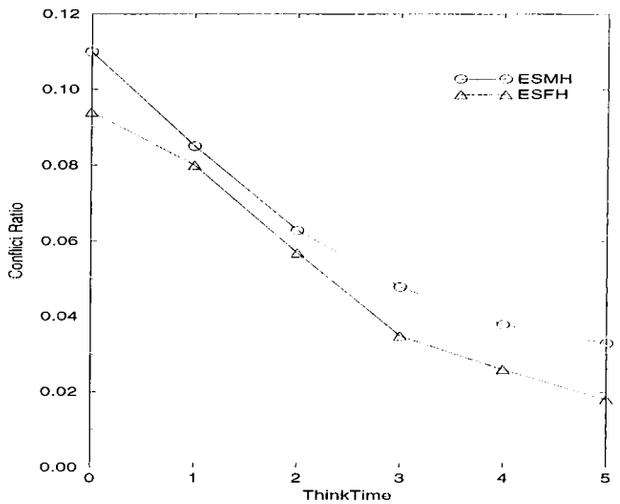


Figure 4.6: Conflict Ratio vs ThinkTime

system, hence resulting in improved performance in terms of the success ratio. In these experiments, we varied the think time from 0 to 5 in increments of 1.

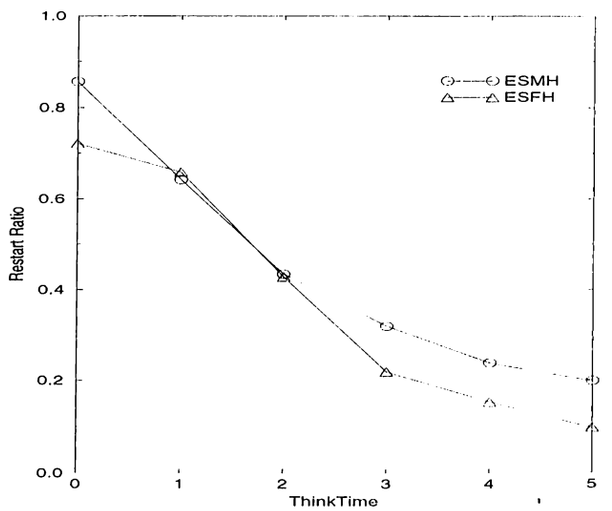


Figure 4.7: Restart Ratio vs ThinkTime

In Figure 4.5, for all parameter settings ESFH performs better, and the performance of both ESMH and ESFH increases at lower ranges of think time, and they are not affected much at higher ranges. The increase at lower levels is due to decreasing contention on the physical resources and data, resulting in less conflict ratio (Figure 4.6) and restart ratio (Figure 4.7). At higher ranges, the contention on the physical resources and the data becomes rather small. Therefore, most of the transactions in case of ESFH succeeds to meet their deadlines. The lower success ratio in case of ESMH is due to wireless link

delay and the less powerful mobile host CPU, as we explained before.

Slack Rate

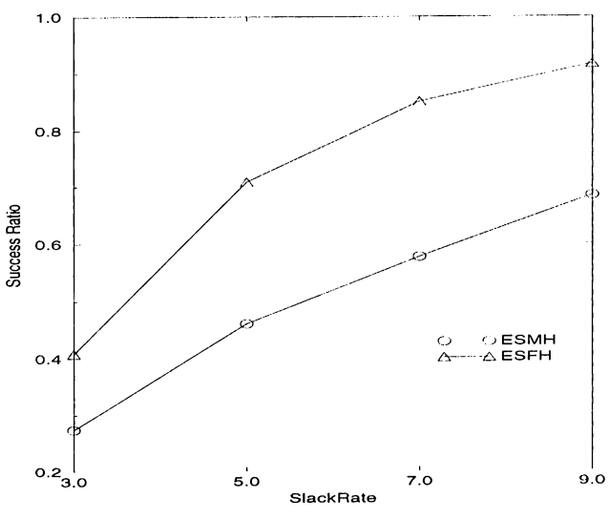


Figure 4.8: Success Ratio vs SlackRate

The slack rate controls the tightness of transaction deadlines. Increasing the slack rate causes the transaction deadlines to become looser, resulting in more transactions to meet their deadlines. In these experiments, we varied the slack rate from 3.0 to 9.0 in increments of 2.0.

As it is shown in Figure 4.8, increasing the slack rate increases the success ratio for both ESMH and ESFH.

Transaction Size

In order to examine the effects of the transaction size, we varied the number of pages accessed by each transaction from 4 to 20 in increments of 4.

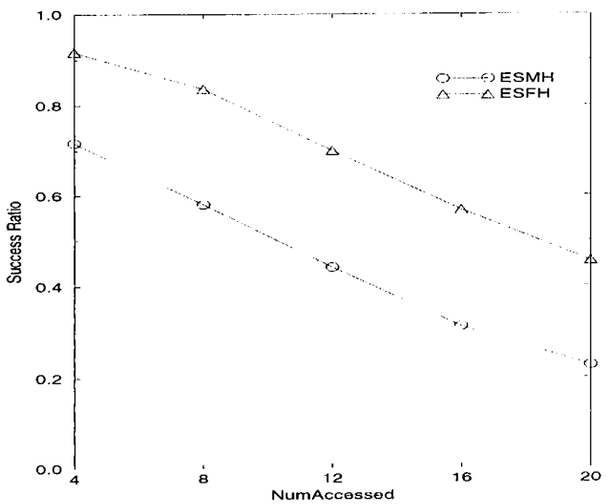


Figure 4.9: Success Ratio vs NumAccessed

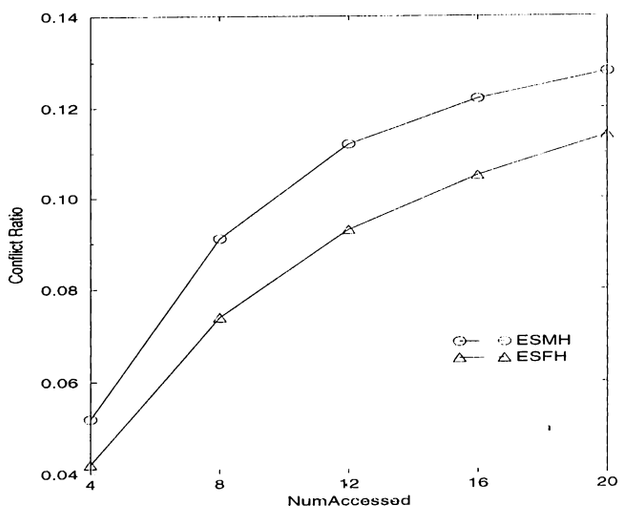


Figure 4.10: Conflict Ratio vs NumAccessed

For both ESMH and ESFH, as the transaction size increases the success ratio decreases linearly as shown in Figure 4.9. This result is due to the fact that the transactions stay longer in the system, resulting in higher data conflict ratio (Figure 4.10), and higher transaction restart ratio (Figure 4.11).

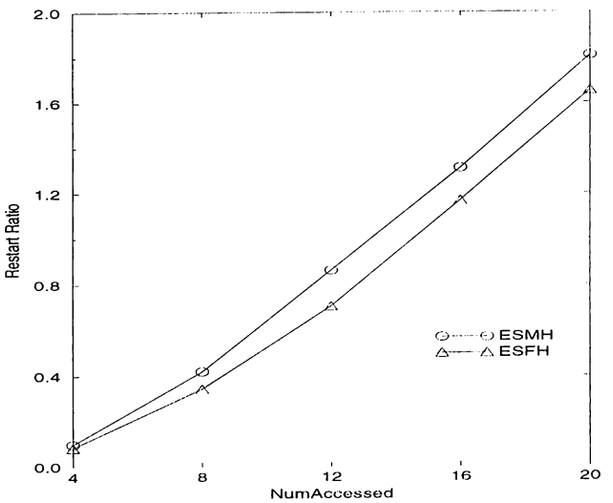


Figure 4.11: Restart Ratio vs NumAccessed

Local Database Size

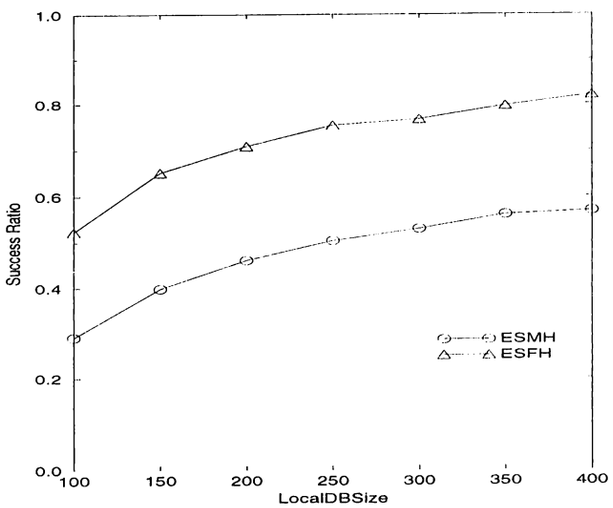


Figure 4.12: Success Ratio vs LocalDBSize

In this experiment, we evaluated the performance of the system by changing the local database size from 100 to 400 pages in increments of 50. The performance results in terms of success ratio is shown in Figure 4.12. Increasing the local database size increases the performance of both ESMH and ESFH. This is due to the decreasing conflict ratio (Figure 4.13), because as the local database size increases the probability that two transactions access the same data item decreases. This also results in smaller restart ratio (Figure 4.14).

Changing the local database size also affects the I/O load of the system.

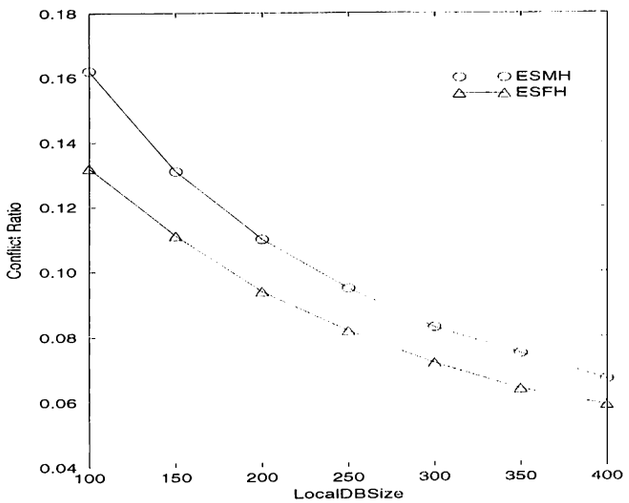


Figure 4.13: Conflict Ratio vs LocalDBSize

When the local database size is increased, the probability that a requested data item can be found in the main memory becomes less. This results in an I/O request to read the the requested data from disk. However, although the I/O load increases, the overall performance of the system becomes better when a larger database size is maintained. This result shows that the number of data conflicts and restarts has more impact on the performance than the amount of I/O operations.

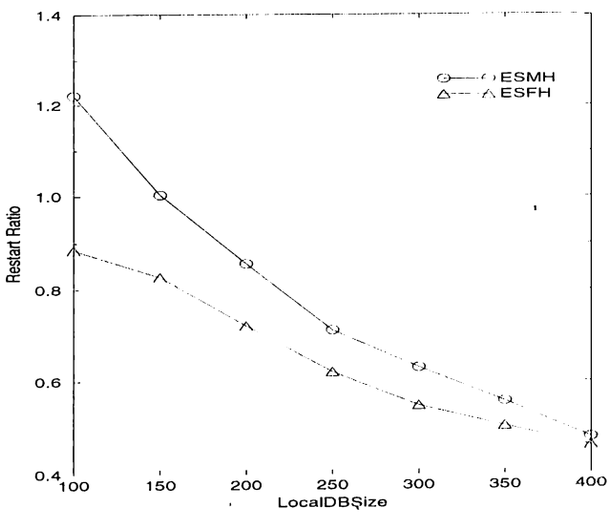


Figure 4.14: Restart Ratio vs LocalDBSize

Data Contention

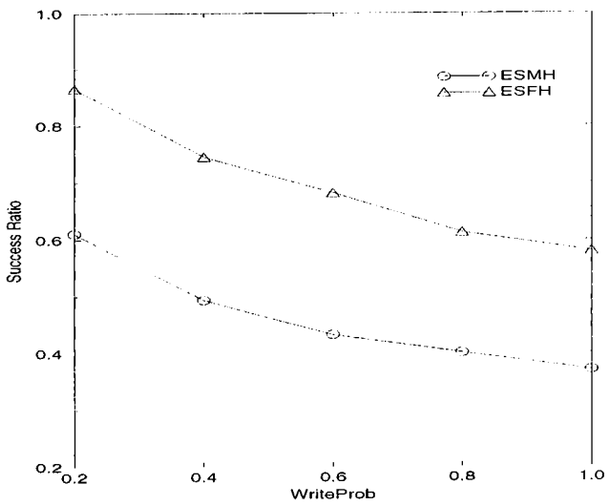


Figure 4.15: Success Ratio vs WriteProb

We evaluated the effect of the data contention by changing the write probability at each data access of an update transaction. As it is shown in Figure 4.15, increasing the write probability degrades the system performance. Larger number of write operations means larger number of data conflicts among transactions (Figure 4.16), and therefore larger number of transaction restarts (Figure 4.17).

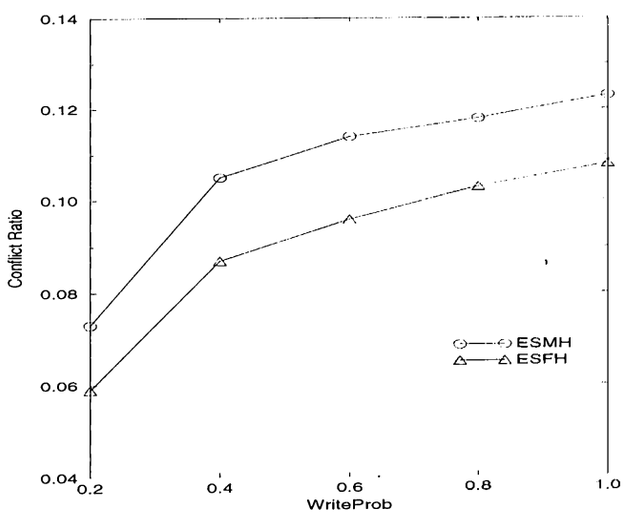


Figure 4.16: Conflict Ratio vs WriteProb

Changing the write probability also affects the I/O load of the system, because each write operation requires a disk access to store the updated data

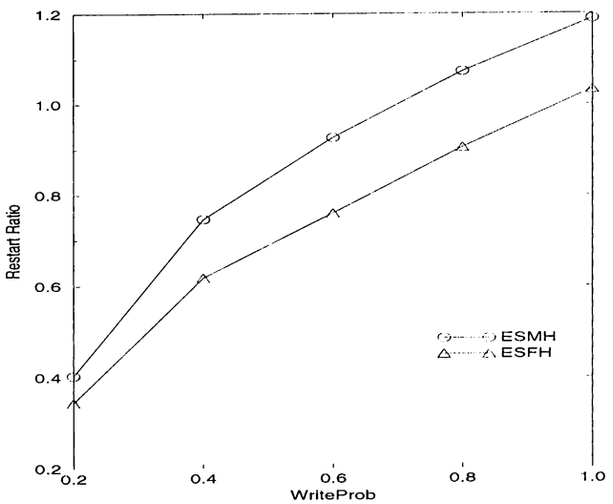


Figure 4.17: Restart Ratio vs WriteProb

item on stable storage. The increased I/O load is another factor that makes the performance worse when the number of write operations is increased.

CPU Availability

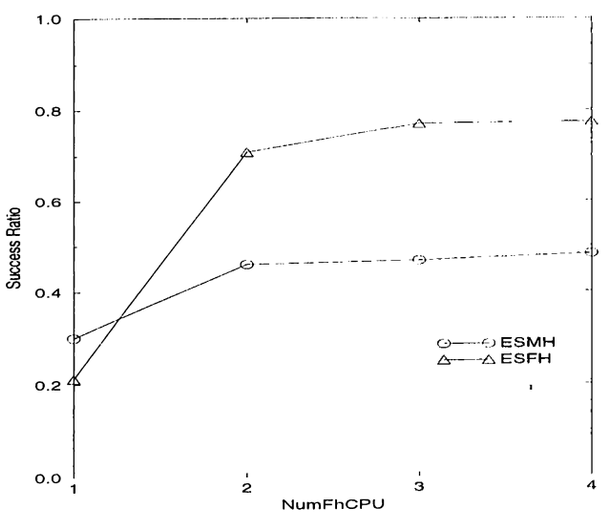


Figure 4.18: Success Ratio vs NumFhCPU

The number of CPUs at a fixed host determines the average load on each CPU. As the number of CPUs increases, the load and contention on each CPU decreases. Especially ESFH is affected by the number of CPUs, because with ESFH, all processing requests for the execution of the transactions are satisfied by the CPUs at the fixed hosts.

Figure 4.18 shows the performance results obtained by varying the number of CPUs at a fixed host. As we can see, the change in the performance of ESFH is more than that of ESMH. At the lowest value of *NumFhCPU*, the performance of ESFH is worse than the performance of ESMH. But at higher values of *NumFhCPU*, due to decreased load on the CPUs, ESFH outperforms ESMH.

User Interaction

A *user interaction* operation is requested from a mobile host which generates transactions. It can be treated as reading a local data item. Therefore, with ESMH a user interaction is satisfied locally, there is no need for any wireless link communication. But, with ESFH, a wireless link delay is experienced.

In this experiment, we varied the number of user interaction operations from 0 to 8 in increments of 2 in order to evaluate the impact of user interaction. Performance results are shown in Figure 4.19.

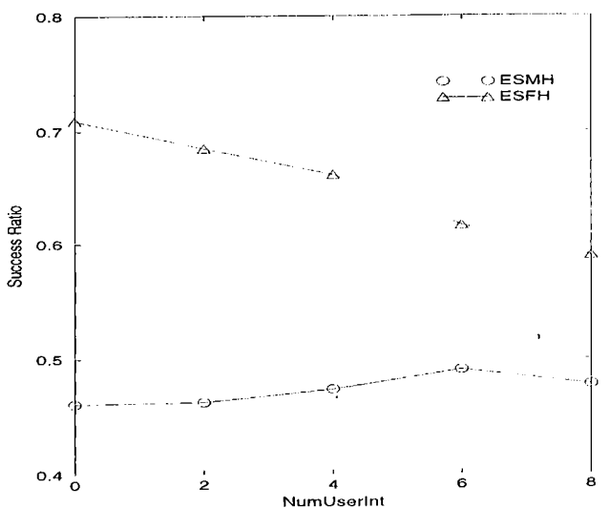


Figure 4.19: Success Ratio vs NumUserInt

As the degree of user interactions is increased the performance of ESMH gets a little bit better. This might be explained by the fact that our system is inclined to assign larger deadlines to transactions when larger number of user interaction operation is expected to be experienced by transactions. The performance of ESFH, on the other hand, degrades as the number of user

interactions increases. This is due to wireless link delay experienced by ESFH for each user interaction request by transactions.

4.2.2 Impact of Mobile System Issues

Disconnection

A mobile host disconnects itself frequently from the system in order to save its limited energy and also due to unavailability of wireless links at some parts. When a mobile host disconnects from the system, it can no longer submit transactions or operations to the system. So, any transaction or operation request by the mobile host cannot be satisfied. Therefore, disconnection results in a degradation in the performance of the system, and this degradation increases as the probability of disconnection increases.

In this experiment, we evaluated the performance of the system by varying the disconnection probability of the mobile host, *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 10 secs.

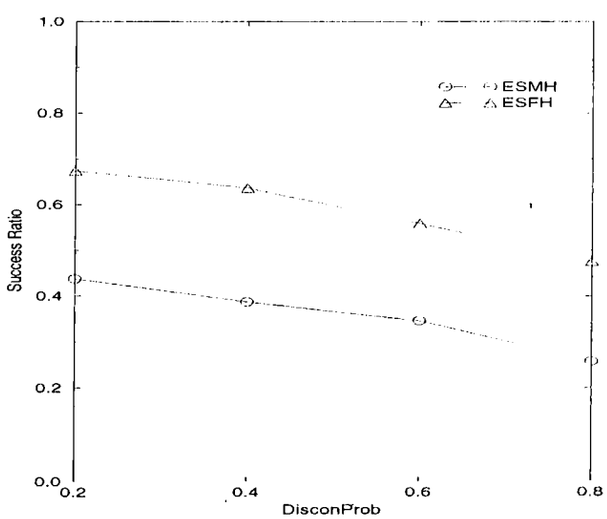


Figure 4.20: Success Ratio vs DisconProb

As it is shown in Figure 4.20, the performance of the system in terms of success ratio decreases while increasing the disconnection probability.

Increasing the probability of disconnections results in decreasing the number of active transactions in the system. Therefore, the number of conflicts between the active transactions decreases (Figure 4.21).

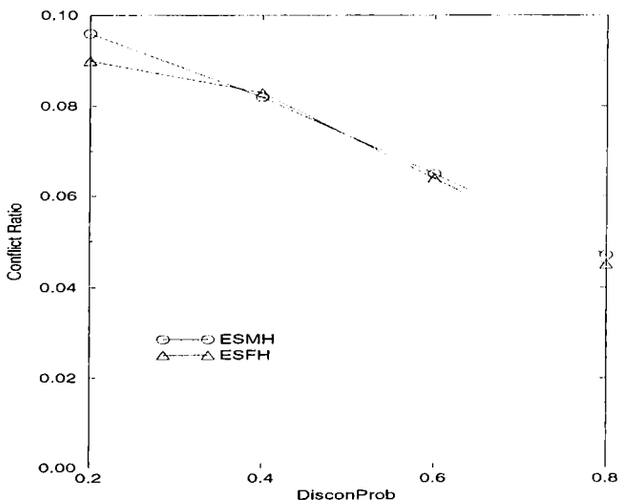


Figure 4.21: Conflict Ratio vs DisconProb

Decreasing the number of active transactions also leads to a decrease in the load of the physical resources. The decrease in both the conflict ratio and the load on the physical resources results in reduced number of restarts (Figure 4.22).

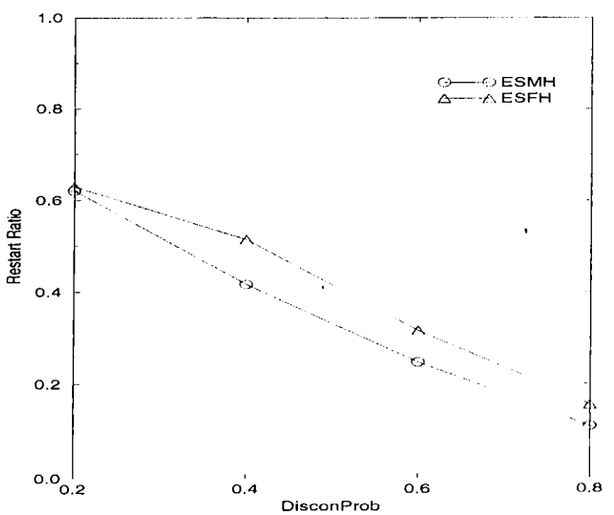


Figure 4.22: Restart Ratio vs DisconProb

Although we observe a decrease in the number of data conflicts and transaction restarts, this does not mean that the performance is improved. While the mobile computers stay disconnected for some period of time, the transactions that require resources on fixed hosts are suspended that makes it difficult to complete their execution within specified deadlines. Therefore, increasing the probability of disconnections results in worse performance.

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 3.4). The communication and search overhead introduced by the handoff process causes more transactions to miss their deadlines.

Remember that a message transmission occurs between the coordinator site and the mobile host during transaction submission/termination and user interaction with ESFH, and for the coordination of *Read*, *Write*, *Commit*, *Abort* operations with ESMH.

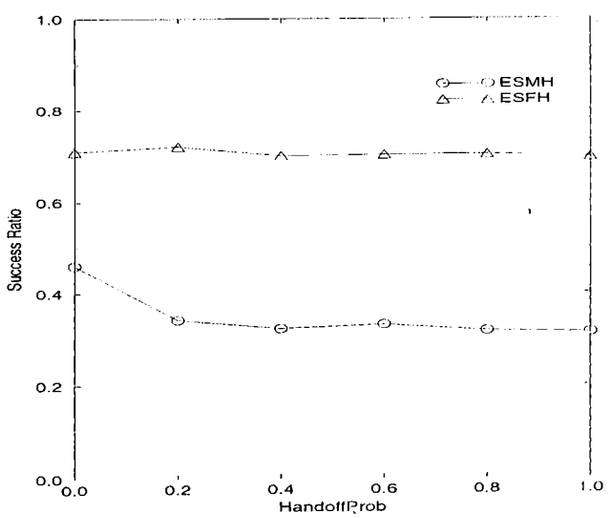


Figure 4.23: Success Ratio vs HandoffProb

The performance results obtained by varying the handoff probability are

shown in Figure 4.23. In this experiment, simulation time is divided into *HandoffInt* time intervals which had the value 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, ESFH is not affected by the handoff process. This is because after the submission of a transaction to coordinator site by a mobile host, there is no need for message transmission between the coordinator site and the mobile host until the transaction completes. But, with ESMH, no handoff case provides better performance than that with handoff case, and for all 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 equal for all values of the handoff probability. Therefore, the average cost of communication and search between the coordinator site and the current MSS of the mobile host is about the same after the system becomes stable. For this experiment, on the average, each transaction experienced about 39 MSS searches and about 15 coordinator site searches during its execution.

In order to see how user interaction affects the system performance with handoff process, we varied the number of user interactions from 0 to 8 in increments of 2. In this experiment, *HandoffInt* had the value of 3 seconds and *HandoffProb* had the value of 0.2.

As illustrated in Figure 4.24, with ESMH, the decrease in the performance of the system due to handoff is not affected by varying the number of user interactions. But, with ESFH, as the number of user interactions increases, the performance degradation of the system also increases. This is due to increasing cost of communication and search for the message transmissions between the coordinator site and and the current MSS. When the number of user interactions is 0, coordinator site search ratio (i.e., average number of coordinator site search requests per transaction) and mobile host search ratio (i.e., average number of mobile host search requests per transaction) have the values of 0.84

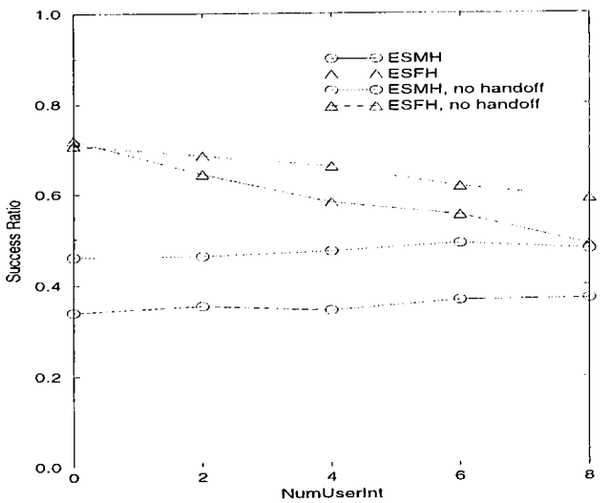


Figure 4.24: Success Ratio vs NumUserInt

and 1.67, respectively. But, these ratios reach the values of 7.84 and 20.40, respectively, when the number of user interactions becomes 8.

Coordinator Relocation

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

In these experiments, we measured the impact of relocating the coordinator site of a mobile host. When a new transaction is submitted to the system, the coordinator is relocated at the current MSS of the generating host of the transaction, if the current MSS is not the coordinator of the generating host. Relocation has a cost for the system that consists of message exchanges between the current MSS and the coordinator of a mobile host. During the transaction execution, the coordinator site remains fixed.

In the first experiment, we measured the performance of the system by varying the handoff probability from 0.2 to 1.0 in increments of 0.2. As we can see in Figure 4.25, relocating the coordinator site degrades the performance of the system for the execution strategy ESFH. This degradation is due to

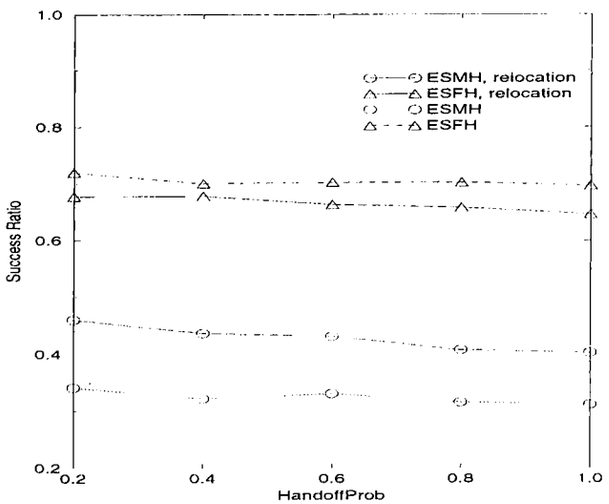


Figure 4.25: Success Ratio vs HandoffProb

cost of the relocation. Because as we explained with the handoff experiment (see Figure 4.23), ESFH is not affected by the handoff operation. Therefore, relocating the coordinator site decreases the performance of the system due to the overhead of relocation operation. On the other hand, with ESMH, the performance of the system increases with relocation, and this increase is nearly the same for all values of the handoff probability. This increase is due to decreasing cost of communication and search between the coordinator site and the current MSS of the mobile host.

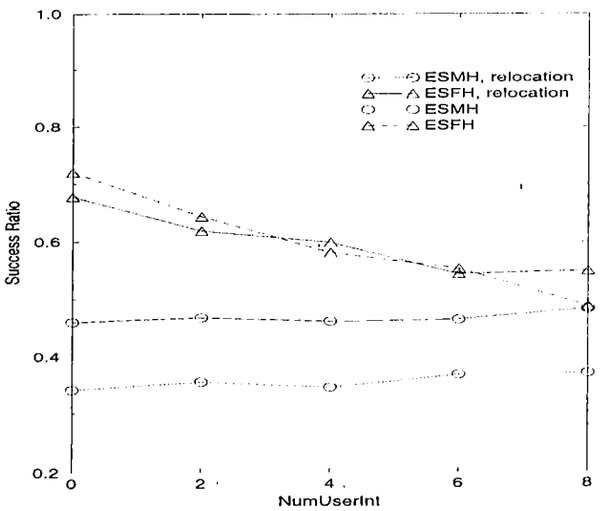


Figure 4.26: Success Ratio vs NumUserInt

In the second experiment, we varied the number of user interactions from 0 to 8 in increments of 2. The performance results are shown in Figure 4.26.

With the execution strategy ESMH, the performance of the system is not affected by changing the number of user interactions, and for all values of the number of user interactions, ESMH with relocation performs better than with no relocation. With ESFH, at lower values of the number of user interactions, relocation does not help the performance of the system to improve due to the cost of the relocation. At higher values, the cost of communication and search outweigh the cost of relocation, so that relocation improves the performance of the system.

Wireless Link Failure

In a mobile computing environment, mobile hosts are more prone to failures than fixed hosts. For mobile hosts, the connection to the fixed part of the network is primarily provided through a wireless link. Wireless links are relatively unreliable.

In this experiment, we measured the performance of the system by varying the failure probability of the wireless link. The simulation time is divided into *FailureInt* time intervals in which the wireless connection of a mobile host to the fixed part of the network is failed with probability *FailureProb* during an interval. Each interval is independent of other intervals.

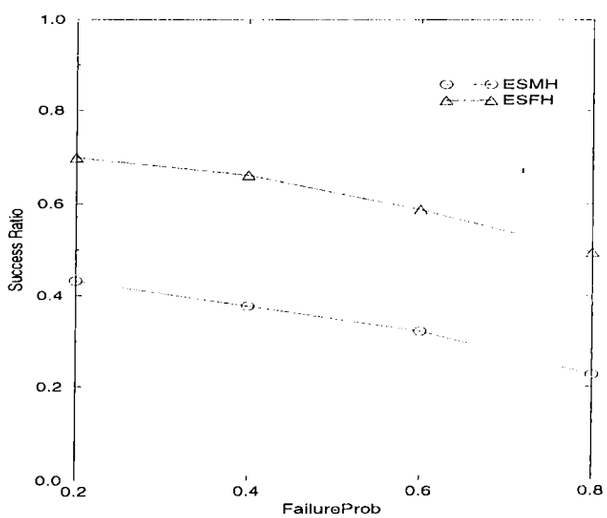


Figure 4.27: Success Ratio vs FailureProb

The performance results for the experiment are shown in Figure 4.27. In

this experiment, *FailureInt* is assigned the value of 5 seconds and the value of failure probability is 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 decreases for both execution strategies ESMH and ESFH.

These results are similar to the performance results we presented with the disconnection experiment at the beginning of this section. This is not surprising, because the only difference between a failure and a disconnection in a mobile environment is that disconnection can be anticipated and necessary actions can be taken beforehand. In this thesis, the only action that we have taken in case of a disconnection is the completion of the transmission of a message that is being transmitted during disconnection.

Chapter 5

Conclusions

The emergence of mobile computing systems with recent developments in wireless communication technology and portable computers enforces the system designers to reconsider the issues in traditional distributed systems. Because, mobile computing systems come with certain constraints that are intrinsic to them. Small size and restricted resource capabilities of mobile computers, limited bandwidth of wireless links, and mobility of users and hosts are some the examples of the constraints of mobile computing systems.

In this thesis, we proposed a mobile database system model that supports real time response to underlying applications. We have constructed a transaction execution model and implemented it on a simulation program in order to evaluate the system performance with two execution strategies, namely, ESMH and ESFH. With the execution strategy ESMH, control information and data request messages for each operation of a transaction is submitted one by one to the coordinator site at the fixed part of the network. Upon completion of processing the request at the fixed network, the result is returned back to the mobile host. All CPU processing of transaction is satisfied at the mobile host that generates the transaction. With the execution strategy ESFH, a transaction is submitted to the coordinator as a whole. The transaction is executed at the fixed host, however, user interaction operation requests of the transaction are handled by the generating mobile host. In this strategy, all the CPU processing of the transaction is satisfied at the fixed part of the network.

We have performed our experiments in two separate parts. In the first part, we evaluated the impact of the system parameters on the performance of the system. In the second part, we evaluated the impact of the mobile system issues on the performance of the system. For all experiments, the performance metric used was the success ratio, i.e., the ratio of transactions that meet their deadlines.

In the first part of the experiments, for all parameter settings, except the lowest value of the number of CPUs at a fixed host in the CPU availability experiment, the execution strategy ESFH performed better than the execution strategy ESMH. This observation is not surprising, because ESMH experiences more wireless link communication in all parameter setting and also uses the less powerful CPU of the mobile host. With ESFH, as the number of user interaction operations increases the performance of the system becomes worse due to increasing number of message transmissions over wireless links. These results show that low-power CPU of the mobile hosts and low-bandwidth wireless links are the bottlenecks of the system.

In the second part of the experiments, it was seen that as the probability of disconnection and wireless link failure increases a degradation occurs in the performance of the system. The performance impact of disconnections and failures does not show much difference. This is because a disconnection is a voluntary failure, and in this thesis the only action we have taken before a disconnection is the completion of the transmission of a message that is being transmitted during disconnection.

With the handoff experiments, when there was no user interaction, the strategy ESFH was not affected by the changes in the handoff probability. However, ESMH performed worse when the probability of handoff was greater than zero; and for all probability values greater than zero, the system performance was about the same. This is because the average cost of communication between the coordinator site and the current MSS of a mobile host is about the same for all positive values of the handoff probability. 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 equal for different values of the handoff probability. Hence, the average cost of communication (i.e., the cost of searches and message transmission delays) does not show much difference.

When we performed handoff experiments by changing the number of user interactions, we saw that the strategy ESMH is not affected by the number of user interactions. This is because, with ESMH, a user interaction operation does not involve any wireless link communication. On the other hand, with ESFH, increasing the degree of user interaction decreases the system performance due to the overhead of wireless link communication.

In another experiment we aimed to investigate the impact of relocating the coordinator site. The purpose of relocating the coordinator site was to reduce the communication cost between the coordinator and the current MSS of a mobile host. In this experiment, we concluded that when the total communication cost between the coordinator site and the current MSS of a mobile host outweighs the cost of relocation, the system performance can be improved. This was the case for all parameter settings with ESMH. However, with ESFH, when the number of user interactions was small, relocating the coordinator site led to worse performance due to the cost of relocation.

The concepts that we have considered in this thesis are the first steps into the field. There are lots of remaining issues that should be considered as a future work. Among such issues are replication, query processing, recovery, and caching of data in mobile environments.

Bibliography

[1] R.K. Abbott, H. Garcia-Molina, "Scheduling Real-Time Transactions: A Performance Evaluation," *ACM Transactions on Database Systems*, vol. 17, no. 3, September 1992.

[2] S. Acharya, R. Alonso, M. Franklin, S. Zdonik, "Broadcast Disks: Data Management for Asymmetric Communication Environments," *ACM SIGMOD Conference*, pp.199-210, 1995.

[3] R. Alonso, H.F. Korth, "Database System Issues in Nomadic Computing," *SIGMOD Conference*, Washington, DC, USA, 1993.

[4] B.R. Badrinath, A. Acharya, T. Imielinski, "Designing Distributed Algorithms for Mobile Computing Networks," *Computer Communications*, 19 (1996), pp.309-320.

[5] D. Barbara, T. Imielinski, "Sleepers and Workaholics: Caching Strategies in Mobile Environments," *ACM SIGMOD International Conference on the Management of Data*, pp. 1-12, 1994.

[6] P.A. Bernstein, V. Hadzilacos, N. Goodman, "Concurrency Control and Recovery in Database Systems," Addison-Wesley, 1987.

[7] O. Bukhres, J. Jing, " Performance Analysis of Adaptive Caching Algorithms in Mobile Environments," *Information Sciences*, 95, 1-27, 1996.

[8] P.K. Chrysanthis, "Transaction Processing in Mobile Computing Environment," *Proceedings of the IEEE Workshop on Advances in Parallel and Distributed Systems*, pp.77-83, Princeton, New Jersey, October 1993.

[9] R. Cáceres, V.N. Padmanabhan, "Fast and Scalable Handoffs for Wireless Internetworks," *Proc. of ACM MobiCom '96*, November 1996.

[10] M.H. Dunham, A.S. Helal, "Mobile Computing and Databases: Anything New?," *SIGMOD Record*, vol. 24, no. 4, December 1995.

[11] A. Elmagarmid, J. Jing, T. Furukawa, "Wireless Client/Server Computing for Personel Information Services and Applications," *SIGMOD Record*, vol. 24, no. 4, December 1995.

[12] G.H. Forman, J. Zahorjan, "The Challenges of Mobile Computing," *IEEE Computer*, vol. 27, 1994.

[13] T. Imielinski, B.R. Badrinath, "Data Management for Mobile Computing," *SIGMOD Record*, 22(1) : 34-39, March 1993.

[14] T. Imielinski, B.R Badrinath, "Mobile Wireless Computing: Challenges in Data Management," *Communications of the ACM*, vol. 37, no. 10, October 1994.

[15] T. Imielinski, S. Viswanathan, B.R. Badrinath, "Data on Air: Organization and Access," *IEEE Transactions on Knowledge and Data Engineering*, vol. 9, no. 3, June 1997.

[16] G. Őzsoyoğlu, R.T. Snodgrass, "Temporal and Real-Time Databases: A Survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 7, no. 4, August 1995.

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

[18] E. Pitoura, B. Bhargava, "Revising Transaction Concepts for Mobile Computing" *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, December 1994.

[19] K. Ramamritham, "Real-Time Databases," *International Journal of Distributed and Parallel Databases*, vol. 1, no. 2, pp. 199-206, 1993.

[20] M. Satyanarayanan, J.J. Kistler, L.B. Mummert, M.R. Ebling, P. Kumar, Q. Lu, "Experience with Disconnected Operation in Mobile Computing Environment," *Proceedings of the 1993 USENIX Symposium on Mobile and Location-Independent Computing*, Cambridge, MA, August 1993.

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

[22] Ö. Ulusoy, "Processing Real-Time Transactions in a Replicated Database System," *International Journal of Distributed and Parallel Databases*, vol. 2, no. 4, 1994.

[23] Ö. Ulusoy, "Real-Time Data Management for Mobile Computing," to appear in *Proc. of the International Workshop on Issues and Applications of Database Technology*, 1998.

[24] Ő. Ulusoy, "Research Issues in Real-Time Database Systems," *Information Sciences*, vol. 87, no. 1-3, 1995.

[25] Ő. Ulusoy, G.G. Belford, "Real-Time Transaction Scheduling in Database Systems," *Information Systems*, vol.18, no.8, pp.559-580, 1993.

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

[27] K.-L. Wu, P.S. Yu, M.-S. Chen, "Energy-Efficient Caching for Wireless Mobile Computing," *International Conference on Data Engineering*, 1996.

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