Abstract

In order to respond to real-time data access requirements of the underlying applications, mobile computing systems need to be supported by real-time DBMSs. In this paper, we discuss critical issues related to real-time transaction management in mobile computing systems. We identify some open research problems and suggest directions for future work on possible solution methods. We also present a mobile database system model that takes into account the timing requirements of applications supported by mobile computing systems. We provide a transaction execution model with two alternative execution strategies for mobile transactions and evaluate the performance of the system considering various mobile system characteristics.

1. Introduction

Recent advances in wireless communication networks and portable computers have led to the emergence of mobile computing systems, in which users with laptops, palmtops, and notebook computers can access to a large number of shared databases. The rapid development in mobile computing technology gives rise to the expectation that, in the near future, tens of millions of users will carry a portable computer with wireless connection to a worldwide information network [10].

In a mobile computing environment, the need for a real-time database management system (DBMS) is strong, because one of the basic requirements in mobile data management is to provide real-time response to transactions of the underlying application. Therefore, a critical item on the agenda for mobile data management research should be to support mobile computing systems with real-time DBMSs. We believe that, the scheduling techniques proposed for real-time DBMSs will find increased interest in the mobile database systems community [17]. In order to make mobile real-time DBMSs viable, research contributions are required in a variety of areas, such as query processing, transaction management, cache consistency, and data replication management. In this paper, we focus our attention on transaction management. In the first part of the paper (Section 2), we provide the description of major problems associated with real-time transaction management in mobile computing systems, together with a discussion on some possible solution methods to those problems. In the second part (Sections 3, 4, and 5), we present a mobile database system model that takes into account the timing requirements of applications supported by mobile computing systems. A mobile transaction execution model constructed for that system is implemented by a simulation program, and the performance of the system is analyzed to evaluate the impact of various mobile system characteristics. Performance results are provided in terms of the fraction of real-time requirements that are satisfied.

2. Real-Time Transaction Management Issues for Mobile Computing

A mobile transaction is distinguished by the feature that the operations of it can be submitted by a portable computer to the data servers from different locations [6]. The distribution of operations implies that communication messages need to be exchanged among data servers to coordinate the execution of these operations. The message delay can substantially impact the transaction response time due to the wireless bandwidth limitation. Transaction execution is also affected by the long delays experienced due to disconnection of mobile computers. Another important source of execution delay is the overhead of concurrency control algorithms that are required to satisfy data consistency requirements of the underlying application. Considering all these delay sources, it can safely be stated that it is quite difficult (if not impossible) to provide schedules that satisfy timing constraints of all the transactions.

Timing constraints of real-time transactions are typically
expressed in the form of deadlines. The **deadline** of a transaction indicates that it is required to complete the transaction before a certain time in the future. A typical categorization of transactions concerns the strictness of the deadlines assigned.

- **Hard deadline transactions** are associated with strict deadlines and the correctness of transaction operations depends on the time at which the results are produced [15]. The system must provide schedules that guarantee deadlines.

- **Soft deadline transactions** are scheduled based on their deadlines, and satisfaction of deadlines is still the primary performance goal in scheduling transactions; however, in this case, there is no guarantee that all deadlines will be met. A soft deadline transaction is executed until completion regardless of whether its deadline has expired or not.

- **Firm deadline transactions** also do not carry strict deadlines, i.e., missing a deadline may not result in a catastrophe, but unlike soft deadline transactions, they are aborted by the system once their deadlines expire. Typically, no value will be imparted to the system if a firm deadline transaction misses its deadline.

As stated before, the basic scheduling goal in a real-time application environment is to meet transaction deadlines. Therefore, a **priority** is assigned to each transaction based on its deadline, and the priorities are taken into account in scheduling the transactions.

Mobile transactions are largely limited to soft or firm deadlines because of the execution delays experienced due to the system constraints mentioned above. As an example application, telecommunication services process soft real-time transactions because a failed call request can be repeated by the subscriber until a successful call is established. However, although the deadlines are not strict, the underlying DBMS is required to process as many transactions as possible within their deadlines (e.g., it is stated in [20] that, in a typical mobile telecommunication system the response times should be bounded for at least 98% of the transactions).

**Concurrency Control**

Serializability is a widely accepted correctness criterion for controlling concurrent execution of transactions in database systems [4]. Serializable schedules provide correct results and leave the database consistent. However, serializability can be restrictive for some mobile applications because of the limitation of concurrency allowed by serializable executions. An extreme approach that can be acceptable in some mobile application environments is to completely eliminate consistency checks while processing transactions. As suggested by Singhal [14], in applications where it is more important to get partially incorrect information quickly than to wait for correct information, a possible approach may be not to exercise any concurrency control and periodically examine the database for inconsistencies and restore it to a consistent state.

In real-time DBMSs, on the other hand, the general approach taken in concurrency control research has been extending traditional concurrency control techniques (that provide a serialization order among conflicting transactions) by applying time-critical scheduling methods to observe timing constraints of transactions. A number of lock-based, optimistic and timestamp-ordering concurrency control protocols have been proposed so far (e.g., [1, 8, 9, 13, 18, 19]). The protocols aim to minimize the number of transactions that miss their deadlines. Adaptation of those concurrency control protocols to mobile database environments needs to be investigated.

**Recovery**

Similar to a conventional distributed DBMS, a mobile DBMS should be able to recover from site and communication failures. Although the mobile environment is more failure-prone than the non-mobile computing environment (due to power restrictions), some of the failures are foreseeable (or voluntary) which enables the system to take special action against such failures [3]. One possible action that can be taken prior to the time of disconnection could be to migrate active transactions to the other computers (probably to non-mobile ones). However, this would lead to a substantial amount of message and data transfer among the computers. In a real-time environment, to reduce this traffic, only the most critical or the highest priority transactions (i.e., the transactions with the stringent timing constraints) can be transferred to the other computers to continue their execution. The other transactions are assumed to afford to wait during the disconnection period without violating their deadlines. Another method to enable high priority transactions to continue their execution when their computer is disconnected is to cache remote data to be accessed by the transactions to the local computer.

It is also possible to increase the reliability of the execution environment for high priority transactions by maintaining their log records at both the local computer and one or more non-mobile remote computers.

Besides transaction priorities, another factor that can be taken into account in controlling the process of transaction logging is the nature of the underlying data. For example, in a mobile telecommunication system there exist two kinds of data: **reference** data and **measurement** data [2]. Reference data represent the configuration of the telecommunication network resources. It is important to maintain the integrity of the reference data, and support is necessary for transactions on this type of data, for logging and data recovery. Measurement data, on the other hand, are collected contin-
uously from the network to identify network problems. The volume of measurement data can be very high and processing log records on this type of data can take a long time which may not be tolerable in a real-time environment. Also, the integrity requirement of this data is not as high as the reference data, and it can be acceptable to lose some portion of it since similar data can again be collected later. Therefore, it can be possible to have multiple levels of integrity and recovery strategies based on the type of the processed data.

3. A Mobile Real-Time Database System Model

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

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

The location of a mobile host can be regarded as a data item that changes as the user crosses the boundary between two cells. Location servers at the fixed hosts are responsible for maintaining the location database and answering location-dependent queries. In order to communicate with a mobile host, it is required to find out its current location. In our mobile system model, forwarding pointers mechanism is used to find out the location of a mobile computer. In this mechanism, when a mobile host changes its location, its new address is deposited at the old location so that messages sent to the mobile host can be forwarded to the new address [7].

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

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

Our transaction execution model is an extension of the model in [16] to mobile computing systems. Each mobile transaction exists in the system in the form of a mobile master process (MMP) at the generating mobile host, a fixed master process (FMP) at the coordinator MSS of the generating host, and fixed cohort processes (FCPs) at sites where the required data pages reside. A fixed transaction, on the other hand, is associated with an FMP at its generating host and FCPs at various sites where it has to access data. Each transaction can have at most one cohort process at a site.

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

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

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

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

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

The execution of a fixed transaction is controlled completely by the FMP of the transaction at its generating host. Each mobile/fixed transaction has a unique priority based on its timing constraint which is used to order resource and data access requests of transactions. Transactions are assumed to carry firm deadlines (i.e., the transaction that has missed its deadline is aborted and discarded from the system). We assume that the system has no apriori knowledge
of the transaction execution requirements, such as the pages to be accessed and the execution time estimation of the transaction.

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

**Concurrency Control**

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

Global serializability is ensured by using the strict two phase locking rule; i.e., by holding the locks of a transaction until after its commitment. The atomic commitment of a transaction is provided by using the Two Phase Commit (2PC) protocol [4].

**Handoff**

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

Our handoff protocol is similar to the one described in [5]. We assume that each MSS broadcasts beacons over its wireless link. Each beacon carries its sender MSS’s address. A mobile host monitors the wireless signal strength it receives from neighboring MSSs. The mobile host may decide to initiate a handoff process when the signal received from a new MSS is substantially stronger than that received from the current one.

**Disconnection**

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

4. Simulation Model

We have implemented our mobile real-time database system model on a simulation program written in CSIM [12]. The simulation model parameters are listed in Table 1. Our mobile computing system consists of NumFHosts fixed hosts, NumMHosts mobile hosts and a fixed network connecting fixed hosts. All fixed hosts are assumed to be MSSs and mobile hosts are connected to the fixed part via wireless links over the MSSs.

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

Table 1. Simulation Parameters

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

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

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

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

Each mobile host in our system has a transaction generator, a transaction manager, a message server, a handoff handler, a disconnection predictor, and an MMP for the mobile transaction executed. Each fixed host has a transaction generator, a transaction manager, a message server, a location server, a data manager, a scheduler, an MMP for each transaction it coordinates, and an FCP for each transaction that has submitted an operation to it.

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

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

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

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

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

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

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

5. Performance Experiments

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ExecStrategy$</td>
<td>ESMH, ESFH</td>
<td>$NumAccessed$</td>
<td>8-16 pages</td>
</tr>
<tr>
<td>$NumFHosts$</td>
<td>10</td>
<td>$NumUserInt$</td>
<td>0</td>
</tr>
<tr>
<td>$NumMHosts$</td>
<td>50</td>
<td>$DiskTime$</td>
<td>12 msec</td>
</tr>
<tr>
<td>$ThinkTime$</td>
<td>0 second</td>
<td>$UpdTrProb$</td>
<td>0.5</td>
</tr>
<tr>
<td>$LocalxDBSize$</td>
<td>200 pages</td>
<td>$WriteProb$</td>
<td>0.5</td>
</tr>
<tr>
<td>$PageSize$</td>
<td>4096 bytes</td>
<td>$SlackRate$</td>
<td>5.0</td>
</tr>
<tr>
<td>$MemSize$</td>
<td>100 pages</td>
<td>$WiredBand$</td>
<td>10 Mbps</td>
</tr>
<tr>
<td>$NumFhCPU$</td>
<td>2</td>
<td>$WirelessBand$</td>
<td>2 Mbps</td>
</tr>
<tr>
<td>$PageCPUTime$</td>
<td>8 msec</td>
<td>$ContMsgSize$</td>
<td>256 bytes</td>
</tr>
<tr>
<td>$MsgCPUTime$</td>
<td>2 msec</td>
<td>$HandoffProb$</td>
<td>0</td>
</tr>
<tr>
<td>$CPURatio$</td>
<td>2</td>
<td>$DisconProb$</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2. Default Parameter Settings

The performance metric used for the evaluations is Success Ratio; i.e., the fraction of transactions that satisfy their deadlines. In order to be able to interpret performance results, we also kept track of the number of conflicting data access requests per transaction and the average number of mobile host/coordinator site search requests per transaction.
In the first experiment, we varied the number of mobile hosts ($NumMHosts$) from 10 to 90 in increments of 20. Increasing the number of mobile hosts corresponds to an increase in the total system load. The range of transaction load obtained in the experiment led to a CPU utilization of 0.13 to 0.72 for the case of ESMH, and 0.34 to 0.87 for the case of ESFH. The ranges for I/O utilization observed with ESMH and ESFH were 0.17 to 0.84 and 0.36 to 0.87, respectively.

The performance results obtained for the execution strategies ESMH and ESFH in terms of the Success Ratio are shown in Figure 1. Remember that ESMH represents the case where the execution site of the mobile transactions is the mobile hosts, and ESFH represents the case where the execution site is the fixed hosts. As it can be seen from the figure, the performance of the system degrades for both strategies ESMH and ESFH as the transaction load increases, and for all ranges of the transaction load ESFH performs better than ESMH. This is because ESMH involves an additional wireless link delay for each data access request, and uses less powerful CPU of the mobile host.

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

The performance results obtained by varying the handoff probability are shown in Figure 2. In this experiment, simulation time is divided into $HandoffInt$ time intervals which has the value of 3 seconds. A handoff process occurs at the beginning of each time interval with probability $HandoffProb$. We varied the handoff probability from 0 to 1 in increments of 0.2. The value of 0 corresponds to the case where no handoff process occurs. In this experiment, we set the value of the number of user interactions to 0.

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

In order to see how the frequency of user interactions affects the system performance with the handoff process, we conducted another experiment. A user interaction operation is issued by a mobile host during the execution of a transaction generated by itself. This operation can be considered as reading a local data item that is required to continue the execution of the mobile transaction. Therefore, with ESMH a user interaction request is satisfied locally, and there is no need for wireless link communication. But, with ESFH, a wireless link delay is experienced between the generating mobile host and the current MSS for satisfying the user interaction requirement of the mobile transaction. To evaluate the impact of user interactions, we varied the number of user interaction operations from 0 to 10 in increments of 2, and set the values of $HandoffInt$ and $HandoffProb$ to 3 seconds and 0.2, respectively.

As illustrated in Figure 3, the performance pattern with
ESMH is not affected by the change in the number of user interactions. The small degradation in performance with both cases (i.e., the cases of with handoff and without handoff) is due to the execution overhead of user interaction operation.

With ESFH, a considerable degradation in the performance is observed as the number of user interactions increases. This result can be explained by the increasing amount of wireless link delay with each additional interaction request during the execution of mobile transactions. The performance degradation of the system with handoff becomes sharper compared to that without handoff. This result is due to the increasing cost of the search for the coordinator site. When the number of user interactions is 0, the average number of coordinator site searches per transaction and the average number of mobile host search requests per transaction have the values of 2.12 and 2.23, respectively. But, these two performance measures reach the values of 10.30 and 28.05, respectively, when the number of user interactions becomes 10.

Another important property of mobile systems that needs to be investigated is disconnection of mobile hosts. A mobile host disconnects itself frequently from the fixed part of the system either to save its limited energy or because of the unavailability of wireless links at some parts. When a mobile host disconnects, it can no longer submit transactions or operations to the system.

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

As expected, the performance of the system in terms of Success Ratio for mobile transactions degrades while the disconnection probability is increased (Figure 4). However, the relative performance of ESMH and ESFH is not affected by the frequency of disconnections.

We conducted some other experiments to evaluate the performance impact of various other mobile system characteristics, such as the coordinator relocation and wireless link failure. The full set of results of all experiments can be found in [11].

6. Conclusions

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

We also performed some experiments to evaluate the impact of the mobile system issues on the performance of the system. The performance metric used in all the experiments
was the Success Ratio, i.e., the fraction of transactions that meet their deadlines. It was observed in the experiments that low-power CPU of the mobile hosts and low-bandwidth wireless links are the bottlenecks of the system. When we evaluated the performance impact of the handoff process, we observed that, with no user interaction, the performance of the strategy ESFH is not affected by the handoff process. Another observation was that, although the handoff process leads to a decrease in the performance with ESMH, increasing the frequency of handoffs does not introduce additional degradation in performance.

References