A Transaction Processing Model for the Mobile Data Access System

K. Segun[1], A.R. Hurson[1], and A. Spink[2]

[1] Computer Science and Engineering Department,
The Pennsylvania State University, University Park, PA 16802, USA
[2] School of Information Sciences and Technologies,
The Pennsylvania State University, University Park, PA 16802, USA

Abstract. Advances in wireless networking technology and portable computing devices have led to the emergence of a new computing paradigm known as mobile computing and a number of applications. As a result, software applications have to be redesigned to take advantage of this environment while accommodating the new challenges posed by mobility.
As mobile users wander about, they are bound to encounter a variety of different information sources (databases) that are often autonomous and heterogeneous in nature. Such a collection of autonomous and heterogeneous database is often known as a multidatabase. The existing multidatabase systems do not readily support mobile computing. A new class of multidatabase that provides access to a large collection of data via a wireless networking connection is proposed — a Mobile Data Access System (MDAS). Within the scope of MDAS, a new transaction-processing model is proposed that allows timely and reliable access to heterogeneous and autonomous data sources while coping with the mobility issue. The proposed model extends the existing multidatabase system without any adverse effect to the preexisting local and global users. This is accomplished through the implementation of multi tiered mobile transaction proxies that manage the execution of mobile transactions on behalf of the mobile user. The proposed transaction-processing model is simulated and the results are analyzed.

1 Introduction

The mobile computing paradigm has emerged due to advances in wireless networking technology and portable computing devices. Mobile computing enables users equipped with portable computing devices to access information services through a shared infrastructure, regardless of physical location or movement. The mobile computing environment is a distributed computing platform with the following differences: the mobility of users and their access devices, frequent disconnection, limited bandwidth and the mobile resource constrains — limited computational and power sources.

Mobile users now have the ability to send and retrieve emails, receive updates on stock prices and weather, and obtain driving directions while in motion using cellular phones, pagers, and PDAs. Wireless transmission media across wide-area tele-

V. Malyshkin (Ed.): PaCT 2001, LNCS 2127, pp. 112–127, 2001.

communication networks are also an important element in the technological infrastructure of E-commerce [21]. The effective development of guided and wireless-media networks will support the delivery of World Wide Web functionality over the Internet. Using mobile technologies will enable users to purchase E-commerce goods and services anywhere and anytime. Naturally, mobile users also desire the same functionality available to them at a stationary computer on a wired network — edit and save changes to documents stored on a file server or to query and update shared data in private or corporate databases. The focus of this paper is on the latter.

As mobile users wander about, they are bound to encounter a variety of different information sources (databases) that are often autonomous and heterogeneous in nature. It would be advantageous if a uniform interface can be presented to the mobile users freeing them from the need to have knowledge of the data representation or data access method employed at different data sources. Organizing a collection of autonomous databases into a multidatabase is therefore desirable. A multidatabase integrates pre-existing autonomous and heterogeneous databases to form a global distributed information-sharing paradigm. To support mobile users, it is necessary to augment the existing multidatabases with wireless networking capabilities. This augmented multidatabase is known as a Mobile Data Access System (MDAS) [13].

The MDAS must have the capability of supporting a large number of mobile users. It is necessary that the MDAS provide timely and reliable access to shared data. Multidatabases have been designed to meet these requirements, albeit within the scope of the fixed networking environment. However, these systems have not been designed to cope with the effects of mobility.

Transactions are the means of access to shared data in databases; this is also the case in a multidatabase and a MDAS. Transaction management in an MDAS environment has some inherent problems due to the full autonomy of local nodes over the execution of transactions and the limitations imposed by the mobile computing environment. In this environment, the global transaction manager (GTM) must be able to deal with: i) different local transaction management systems; ii) different local concurrency control mechanisms; iii) lack of communication with local nodes, and iv) limitations of the mobile computing environment.

Concurrency control is needed in order to increase throughput and to allow timely and reliable access to shared data and must therefore support simultaneous execution and interleaving of multiple transactions. In an MDAS environment, the concurrency control algorithm has to overcome the effects of the local autonomy, in addition to constraints imposed by the mobile units.

As an example, consider a transaction in execution on a stationary computer on a wired network. The occurrence of a disconnection is often treated as a failure in the network thus, when this occurs the executing transaction is aborted. In a mobile computing environment, which is characterized by frequent disconnection (users may choose to disconnect voluntarily, for instance to conserve battery life), disconnection cannot be treated as a failure in the network.

Transactions issued from mobile clients tend to be long-lived. Thus, transactions issued by mobile users are exposed to a larger number of disconnections. Another effect of long-lived transactions is that it could result in low system throughput. Long-lived transactions are more likely to result in conflicts. Pessimistic locking schemes in the implementation of concurrency control could result in blocking of concurrently executing transactions, resulting in deadlocks and aborted transactions. On the other

hand, employment of optimistic concurrency control could result in a high rate of transaction restarts. Thus, a new transaction model is needed for the MDAS environment that manages concurrency control and recovery, handles frequent disconnection, and address the issue of long-lived transactions while at the same time does not violate the autonomy of the local data sources.

The goal of this paper is to present such a transaction processing model. The model is built on the concept of global transactions in multidatabase based on the Summary Schemas Model [6]. This work expands our effort reported in [13] by implementing an additional layer on top of the MDBS that handles mobile transactions, disconnection, and long-lived transaction.

Section 2 addresses the background material on multidatabase systems and mobile computing and the issues that affect the MDAS. Section 3 is a description of the MDAS transaction processing model and the necessary protocols. Section 4 presents the results and analysis of the simulation model of the proposed model. Finally, Section 5 concludes the paper and addresses several future research issues.

2 Background

The basis of the MDAS is the mobile computing environment and the multidatabase. Thus, this section gives a brief overview of the mobile computing environment, multidatabase systems, and the concepts and issues that characterize these environments.

2.1 Mobile Computing Environment

The mobile computing environment is a collection of mobile hosts (MH) and a fixed networking system [8],[10],[13]. The fixed networking system consists of a collection of fixed hosts connected through a wired network. Certain fixed hosts, called base stations or Mobile Support Stations (MSS) are equipped with wireless communication capability. Each MSS can communicate with MHs that are within its coverage area (a cell). MHs can move within a cell or between cells, effectively disconnection from one MSS and connecting to another. At any point in time, a MH can be connected to only one MSS. MHs are portable computers that vary in size, processing power, and memory. Wireless Communication, mobility, and portability are three essential properties of mobile computing that pose difficulties in the design of applications [10].

2.2 Multidatabase Systems

A multidatabase system integrates pre-existing local databases to form a single integrated global distributed database system. It is a collection of autonomous local database systems (LDBS), possibly of different types. The integration of the DBMSs is performed by multiple software sub-systems at the local databases [3],[19]. The local databases are unaware of the existence of the global database [20]. Local

autonomy is the key requirement in the design of a multidatabase. In a multidatabase there are two types of users: local users and global users. Local autonomy guarantees that the local users access their own local database independent of, and unaffected by, the existence of the multidatabase and its global users. Autonomy in multidatabases comes in the form of; design autonomy, participation autonomy, communication autonomy, and execution autonomy [3].

2.3 MDAS Issues

The MDAS is a multidatabase system that has been augmented to provide support for wireless access to shared data. Issues that affect multidatabases are therefore applicable to the MDAS. Mobile computing raises additional issues over and above those outlined in the design of a multidatabase. In the following we examine the effects of mobility on query processing and optimization, and transaction processing.

- **Query Processing and Optimization**: The higher communication cost of wireless medium and limited power of a mobile unit may lead to the design of query processing and optimization algorithms that focus on reducing the financial cost of transactions and consideration for query processing strategies for long-lived transactions that do not rely on frequent short communications but longer communications. Query optimization algorithms may also be designed to select plans based on their energy consumption. Approximate answers will be more acceptable in mobile databases than in traditional databases due to the frequent disconnection and the long latency time of transaction execution [1].
- **Transaction Processing**: Since disconnection is a common mode of operation, transaction processing must provide support for disconnected operation. Temporary disconnection should be tolerated with a minimum disruption of transaction processing, and suspension of transactions on either stationary or mobile hosts. In order for users to work effectively during periods of disconnection, mobile computers will require a substantial degree of autonomy [1],[13],[18]. Effects of mobile transactions committed during disconnection should be incorporated into the database while guaranteeing data and transaction correctness upon reconnection [18]. Atomic transactions are the normal mode of access to shared data in traditional databases. Mobile transactions that access shared data cannot be structured using atomic transactions. However, mobile computations need to be organized as a set of transactions some of which execute on mobile hosts and others that execute on the mobile support hosts. The transaction model will need to include aspects of long transaction models and Sagas. Mobile transactions are expected to be lengthy due to the mobility of the data consumers and/or data producers and their interactive nature. Atomic transactions cannot satisfy the ability to handle partial failures and provide different recovery strategies, minimizing the effects of failure [1],[7],[20].
- **Transaction Failure and Recovery**: Disconnection, bandwidth limitations, and higher probability of damage to the mobile devices are some of the possible sources of failure in mobile environments. Special action can be taken on behalf of active transactions at the time a disconnection is predicted — a transaction processes may be migrated to a stationary computer particularly if no further user interaction is required. Remote data may be downloaded in advance of the

predicted disconnection in support of interactive transactions that should continue to execute locally on the mobile machine after disconnection. Log records needed for recovery may be transferred from the mobile host to a stationary host [1].

2.4 Summary Schemas Model

The Summary Schemas Model (SSM) has been proposed in [6] as an efficient means to access data in a heterogeneous multidatabase environment. The SSM uses a hierarchical meta structure that provides an incrementally concise view of the data in the form of summary schemas. The hierarchical data structure of the SSM consists of leaf nodes and summary schema nodes. The leaf nodes represent the portion of local databases that are globally shared. Each higher-level node (summary schema nodes) provides a more concise view of the data by summarizing the semantic contents of its children. The terms in the schema are related through synonym, hypernym and hyponym links. The SSM allows a user to submit a request in his/her own terms. It intelligently resolves a query into a set of subqueries using the semantic contents of the SSM meta data. The overall memory requirements for the SSM, compared to the requirements of a global schema, are drastically reduced by up to 94%. Subsequently, the SSM meta data could be kept in main memory, thus reducing the access time and query processing time. Furthermore, for resource scares MDAS access devices, caching the upper levels of the SSM meta data structure allow a great amount of autonomy to each mobile unit. Finally, the SSM could be used to browse data by "stepping" through the hierarchy, or view semantically similar data through queries.

3 Proposed Transaction Processing Model

The proposed MDAS transaction model is based on a multi tiered approach capable of supporting pre-existing global users on the wired network in addition to mobile users. The proposed transaction model is implemented as a software module on top of the pre-existing multidatabase management system. Integration of the mobile computing with the pre-existing multidatabase system in then the key challenge in MDAS.

3.1 MDAS Transactions

We may distinguish three types of transactions:

- Local transactions that access only local data at each LDBS,
- Global Transactions that access data at more than one LDBS, and
- Mobile transactions that could access data from more than one LDBS.

In reality, a mobile transaction is no different from a global transaction as far as the MDBS layer is concerned. However, a number of factors make it sufficiently different enough to consider it as a separate transaction type in the MDAS:

- Mobile transactions require the support of stationary hosts for their computations and communications.

- Mobile transactions might have to split their computations, with one part executing on a mobile client and the other part executing on a stationary host.
- Mobile transactions might have to share state and data. This is a violation of the revered ACID transactions processing assumptions.
- Mobile transactions might have to be migrated to stationary hosts in order to accommodate the disconnection of the mobile client.
- Mobile transactions tend to be long lived. This is a consequence of the frequent disconnection experienced by the mobile client and the mobility of the mobile client.

3.2 MDAS Transaction Model

The MDAS as we envision it, consists of a software module, called a Mobile Transaction Manager (MTM), implemented above the MDBS layer. The two layers combined form the MDAS. The MTM is responsible for managing the submission of mobile transactions to the MDBS layer and their execution. Thus, the MTM acts as a proxy for the mobile unit, thereby establishing a static presence for the mobile unit on the fixed network. The other half, the GTM is responsible for managing the execution of global transactions submitted by non-mobile users and mobile transactions submitted on behalf of the mobile unit by the MTM.

Our approach is based on the principle that the computation and communication demands of an algorithm should be satisfied within the static segment of the system to the extent possible [2]. In another words, we attempt: i) to localize communication between a fixed host and a mobile host within the same cell, ii) to reduce the number of wireless messages by downloading most of the communication and computation requirements to the fixed segment of the network, and iii) to develop distributed algorithm based on the maintained logical structure among the fixed network.

Mobile transactions are submitted to the MDBS layer in a FIFO order by the MTM. There are two operating modes that reflect the level of delegation of authority to the proxy by the mobile client.

- **Full Delegation Mode**: In this mode the mobile client delegates complete authority of the mobile transaction to the MTM. The MTM has the authority to commit the transaction upon completion. If there is a conflict the MTM may decide to abort the transaction and resubmit it, later on. In any case, the mobile client is notified of the status of the transaction and will receive the results (if any).
- **Partial Delegation Mode**: In this mode more participation is required of the mobile client. The mobile client has the final say on whether or not the transaction should be committed. The MTM submits the transaction to the MDBS and manages its execution on behalf of the mobile client. Upon completion of the operations of the transaction, the mobile client is notified and the MTM waits for the commit or abort message from the mobile client.

In applying the proposed transaction-processing model to the MDAS we may derive the following benefits:

- Our protocol decouples the effects of mobility from the MDBS. Hence, any developed concurrency control and recovery mechanism can be readily adopted into our protocol.

- The MDBS layer does not need to be aware of the mobile nature of some nodes. The mobile transactions are submitted to the MDBS interface by the transaction proxies. The MDBS interacts with the transaction proxy as though it were the mobile unit. In the case of a mobile transaction, most of the communication is within the fixed network and as far as the MDBS is concerned, a static host has initiated the transaction.
- The operations of non-mobile users are unaffected by the transactions of mobile users. The effects of long-lived transactions can be effectively and efficiently handled. Delegating the authority to commit and/or abort a transaction on behalf of the mobile host to the transaction proxy can minimize the effects of long-lived transactions. Thus, transactions initiated by non-mobile users will experience less conflict and as a consequence system throughput and response times are not severely affected.
- The mobile host may disconnect and freely change location since the transaction proxy acts on its behalf without requiring any participation from the mobile host unless it is interested in the outcome.

3.3 Operating Modes

Mobile Host – MSS Relationship. In the proposed MDAS transaction-processing model, communication occurs through the exchange of messages between static and/or mobile hosts. In order to send a message from a mobile host to another host, either fixed or mobile, the message is first sent to the local MSS over the wireless network. The MSS forwards the message to the local MSS of the other mobile host, which forwards it over the wireless network to the other mobile host if it is meant for a mobile host. Otherwise, the message is directly forwarded to the fixed host. The location of a mobile host within the network is neither fixed nor universally known in the network. Thus, when sending a message to a mobile host the MSS that serves the mobile host must first be determined. This is a problem that has been addressed through a variety of routing protocols (e.g. Mobile IP, CDPD) at the network layer [4,11]. We are not concerned with any particular routing protocol for message delivery but instead assume that the network layer addresses this issue.

Each MSS maintains a list of *id*s of mobile hosts that are local to its cell. When a mobile host enters a new cell, it sends a *join message* to the new MSS. The *join message* includes the *id* (usually the IP address) of the mobile host. When the MSS receives the *join message* adds the mobile host to its list of local mobile hosts. To change location, the mobile host must also send a *leave message* to the local MSS. The mobile host neither sends nor receives any further messages within the present cell once the *leave message* has been sent. When the MSS receives the *leave message* from the mobile host, it removes the mobile host *id* from its list of local mobile hosts.

Disconnection is often predictable by a mobile host before it occurs. Therefore, in order to disconnect, the mobile host sends a *disconnect message* to the local MSS. The *disconnect message* is similar to the *leave message*, the only difference being that when a mobile host issues a *leave message* it is bound to reconnect at some other MSS at a later time. A mobile host that has issued a *disconnect message* may or may not reconnect at any MSS later. When the MSS receives the *disconnect message* a *disconnect flag* is set for the particular mobile host id. If an attempt is made to locate a

disconnected mobile host the initiator of the search will be informed of the disconnected status of the mobile host.

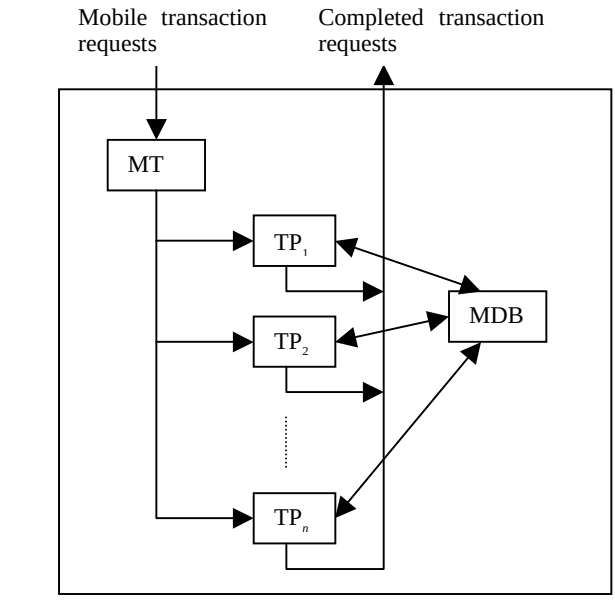
A mobile host that issues a *leave message* or a *disconnect message* must issue a *reconnect message* to reconnect to a MSS. The *reconnect message* must include the *id*s of the mobile host and the previous MSS at which it was last connected. The *id* of the previous MSS is necessary so that the new MSS and the previous MSS can execute any handoff procedures necessary, for instance, unsetting the *disconnect flag*. When the MSS receives the *reconnect message* it adds the mobile host to its list of local mobile hosts and executes any handoff procedures with the prior MSS.

Mobile Host – MTM Relationship. To initiate a transaction, the mobile host sends a *Begin-Transaction* message to the MTM. The MTM acknowledges the request by returning a transaction sequence number. Each MSS has a MTM associated with it and transaction sequence numbers are assigned in a distributed manner among the MTMs in the system using any distributed ordering algorithm, for example, Lamport's algorithm [12]. The mobile host tags each transaction request message with a transaction *id*, which is composed of the mobile host *id*, and the transaction sequence number. The transaction request message is composed of the mobile host *id*, the transaction sequence number, and the transaction operations. To signify the completion of a transaction request, an *End-Transaction message* is sent to the MTM. Transaction execution is delayed until the receipt of the *End-Transaction message*. This is in order to guarantee that the entire transaction as a whole is submitted to the MDBS.

3.4 Transaction Processing Model Work Flow

The transaction processing model workflow can be described as shown in Fig. 1.

- The mobile host initiates a transaction request message. The message is received by the MSS, and is forwarded to the associated MTM.
- The MTM receives the transaction request from the MSS. The transaction request is logged and the transaction *id* (transaction sequence number + mobile host *id*) is placed in the ready list. A transaction proxy is created to execute the transaction.
- The transaction proxy removes a transaction *id* from the ready list and inserts it into the active list. The transaction proxy translates the transaction request and then submits the transaction to the MDBS for execution.
- The transaction request is executed at the MDBS layer and the results and/or data are returned to the transaction proxy.
- The transaction proxy places the transaction *id* in the output list along with the results and data to be returned to the mobile host.
- The MTM initiates a search for the location of the mobile host and the results are transferred to the mobile host if it is still connected and then the transaction *id* is removed from the ready list.



MTM: Mobile Transaction Manager TP: Transaction Proxy
MDBS: Multidatabase System

Fig. 1. Transaction processing workflow

3.5 Disconnected Operation

We turn our attention to the case where the mobile host is no longer connected to the local MSS while the transaction is still in execution. By handing over transaction execution to transaction proxies, the disconnection of a mobile host or its relocation does not affect the transaction execution. The key issue to be addressed is how to notify the mobile host of the results of the transaction execution. In this case the following actions are taken:

On reconnection at the new MSS the mobile host should supply the *id* of the previous MSS to which it was connected. A handoff procedure is then initiated between the two MSSs.

- As part of the handoff procedure, the MTM at the previous MSS searches its ready list, if the transaction request issued by the mobile host has not yet been processed it is forwarded to the MTM at the new MSS and inserted into its ready list. Thus, control of transaction execution is transferred to the new MSS.
- If the transaction has completed its execution then the results are forwarded to the new MSS, which subsequently returns them to the mobile host.
- If the transaction is still active then control is not transferred but the new MSS places the transaction request in its active list but marks it as being executed at another site. The previous MSS will initiate a search for the new MSS of the mobile host when the transaction is complete in order to transfer the results to it.

4 Simulation Result and Analysis

4.1 Simulator

A simulator was developed to measure the feasibility of the proposed protocol within an MDAS environment. The MDASsim simulator and some results from a comparison between the two modes of operation of the Mobile Transaction Manager (Full Delegation Mode and Partial Delegation Mode) are presented. The simulator is based on the DBsim simulator presented in [15]. However, DBsim has been extended to support the concepts of a multidatabase and the MDAS.

The DBsim is an event driven simulator written in C++. The DBsim was designed as a framework to simulate different scheduling policies. Its architecture is based on an object-oriented paradigm, with all the major components implemented as classes in C++. The simulator is a collection of cooperating objects, comprising of: the event controller, transaction manager (TM), scheduler, data manager (DM), and the bookkeeper. A multidatabase is much more complex to model, compared to a distributed database mainly due to the local autonomy and heterogeneity issues. As a result, the DBsim was enhanced with additional flexibility to simulate the important aspects of the MDAS environment. In order to achieve this we introduced three new concepts to the simulation model:

- The DBsim architecture implemented a single transaction manager. We have departed from the single transaction manager module implemented in the original DBSim simulator by allowing a transaction manager at each of the local nodes.
- An additional layer above the local transaction managers was implemented to manage global and mobile transactions. This is the global transaction manager (GTM). For the purpose of our simulation, the GTM serves as the Mobile Transaction Manager (MTM) as well.
- We have introduced the concepts of global and mobile transactions into the simulation model.
  For each simulated local node we have one data manager (DM) object, one scheduler object and one transaction manager (TM) object. The GTM object is responsible for creating mobile and global subtransactions that generate operations to the transaction managers at each local node. The architecture of our simulator is shown in Fig. 2.

4.2 Global Transaction Manager

A Global Transaction is resolved by the summary schema's meta data. As a result, the global transaction is decomposed into several subtransactions, each resolved at a local node. This process also recognizes a global transaction manager for the global transaction — a global transaction manager is the lowest summary schema node that semantically contains the information space manipulated by the transaction. In our simulated environment, the number of local nodes at which the transaction is resolved is chosen randomly from the number of nodes in the system. The global or mobile transaction makes calls to the local transaction managers to begin execution of the subtransactions.

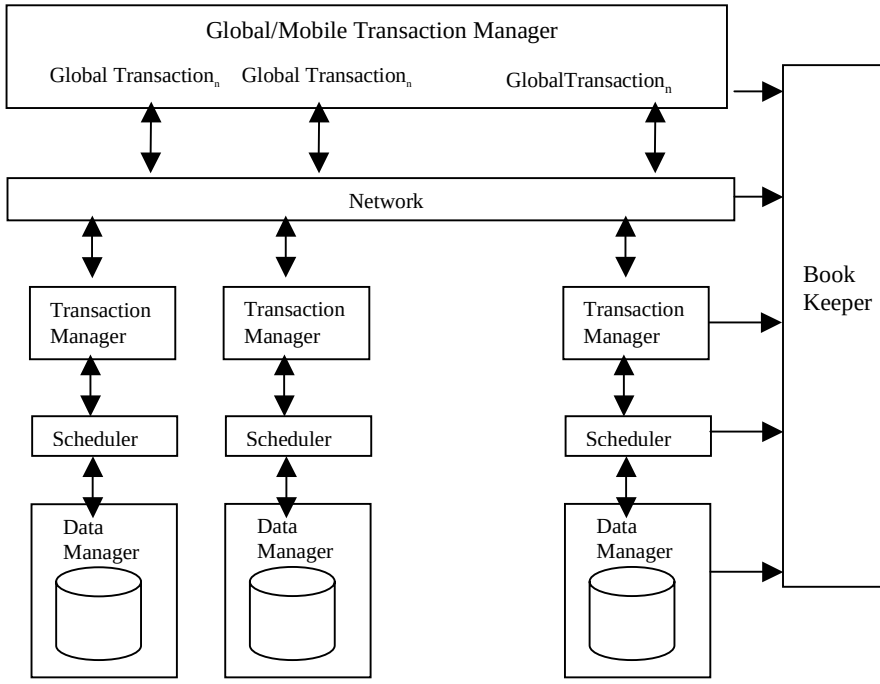


Fig. 2. MDASSim architecture

To allow multiprogramming (MP) level at each local node, the simulator maintains a fixed number of local transactions executing simultaneously at each node — this number is varied for different simulation run. A fixed number of global/mobile transactions are also maintained in the system. The ratio of global to mobile transaction is varied for different simulation runs, as well. Together, the fixed number of simultaneous local transactions and the fixed number of global/mobile transactions serve as an approximation of a system with a constant load. Every time a transaction (local, global or mobile) is terminated, a new one is created after some random delay.

Upon creation (submission) of a transaction (subtransaction) to a local node, its operations (read, write, commit or abort) are begun to schedule for execution. Every time an operation finishes successfully, the transaction, after a short delay, generates a new operation or it decides end the transaction by sending a commit or abort operation.

4.3 Commit Protocol

Each local node implements the two-phase commit (2PC) protocol. In case of a global or mobile subtransaction, the GTM coordinates the commit protocol so as to ensure that either all or none of the subtransactions succeed to preserve the atomicity of the global transaction. A timeout is used to simulate the obtaining of permission to commit the mobile transaction from the mobile unit when there is a need to do so. A

commit or abort is returned based on the probability of communication between the MTM and the mobile unit during the timeout period.

4.4 Simulation Parameters

The behavior of our multi-tiered control level protocol is determined based on several parameters. Some of these parameters are hardware, software, and administrative dependent, and others are application dependent. It is very important that reasonable values be selected for each parameter. The system parameters were derived from the underlying platform. Additional parameters for the mobile component of the system were obtained from the work reported in [13]. These parameters are presented in Tables 1-3.

Table 1. Min and max values of interval parameters

Parameter	Min.	Max.
Number of operations in local transactions selected	2	8
Number of operations generated in a burst	3	5
Time between transactions	10ms	100ms
Time between operation requests	1ms	10ms
Time between operations in a burst	1ms	3ms
Time to perform a disk operation	8ms	16ms
Restart delay	500ms	1500ms

Table 2. Application parameters

Parameter	Value
Number of transactions	20000
Size of address space, # of resource units	20000
Hot spot size, # resource units	2000
Hot spot probability	50%
Abort probability	0.1%
Read probability	80%
Burst probability	20%
Block size	4KB

Table 3. Global and Mobile Unit Parameters

Parameter	Default Value
Number of global/mobile transactions in the systems	10
Service time for each communicated message to the mobile unit selected randomly	0.3 – 3 sec
Probability of mobile unit not being found after the timeout	0.20

4.5 Simulations and Results

Our simulations were based on a constant load. The MPL (the number of simultaneous local transactions) at the local sites during each simulation run was constant (varied from 1 to 25) with a mix of global and mobile transactions. At all times, the total number of global transactions (global/mobile) is also constant while the ratio of global to mobile transactions varies for each simulation run (chosen as 20%, 40%, 50%, 60% and 80%). The throughput was used as the performance measure and it was measured against parameters such as: number of simultaneous local transactions (MP-Level), the varying ratio of global to mobile transactions, and the two different operating modes of the MTM (full-delegation and partial delegation modes). In general, as one could expect, at a lower MP-level, the global/mobile throughput was higher due to the fewer local transactions in the system and less likelihood of conflicts among transactions. As the MP-level increased, the global/mobile throughput dropped as a result of more local transactions in the system with increased likelihood of indirect conflicts among global/mobile transactions.

Figs 3 – 5 show the throughput of both global and mobile transactions as the number of simultaneous local transactions and the ratio of global to mobile transactions are varied. The charts compare the results under the Full Delegation mode of operation and the Partial Delegation mode of operation. As can be noted, the performance under the Full Delegation mode (FDM) surpasses that of the Partial Delegation mode (PDM) since the proxy needs to communicate with the mobile unit under the latter scheme. However, such performance degradation is quite tolerable specially, when one considers the flexibility and adaptability of our approach.

5 Conclusion and Future Directions

5.1 Conclusion

This paper proposed a new transaction-processing model for the mobile data access system (MDAS). The proposed multi-tiered transaction model uses the concepts of transaction proxies to manage the execution of mobile transactions. To provide support for mobile transactions, a layer, the Mobile Transaction Manager (MTM), is implemented above the pre-existing multidatabase system. Using proxies the proposed model decouples the effects of mobility – frequent disconnection, limited bandwidth, limited computational resources, etc. – from the multidatabase systems.

Two modes of operation, namely, Full delegation mode and partial delegation mode, were proposed to address the level of participation of a mobile unit in the completion of a mobile transaction. In the Full Delegation mode of operation, the mobile unit relinquishes control of the final commit/abort of a transaction to the MTM. In the Partial Delegation mode of operation, the mobile unit has the final say on whether to commit or abort the transaction. The MTM must communicate with the mobile unit when the transaction is ready to be committed. However, should the mobile unit be unavailable, the MTM is free to abort the transaction after a sufficient time out period.

A simulator written in C++ was developed to evaluate the feasibility and performance of the proposed transaction-processing model. The simulation results

showed that the performance of the Full Delegation mode of operation is better than the Partial Delegation mode. This comes about as a result of fewer communications between the mobile unit and the multidatabase system. The performance of the system was evaluated by varying the number of simultaneous local transactions executing at each node and by varying the ratio of global to mobile transactions present in the system.

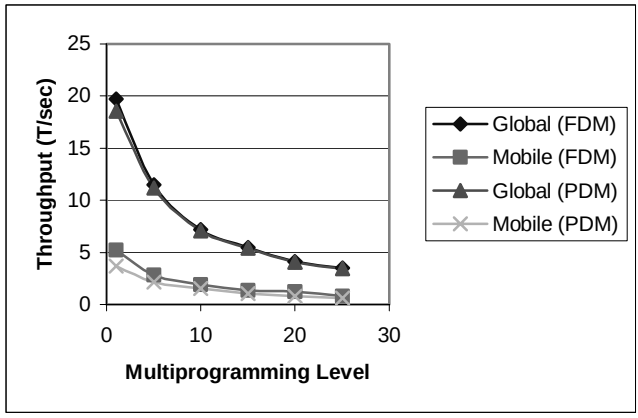


Fig. 3. Throughput with 20% Mobile Transactions

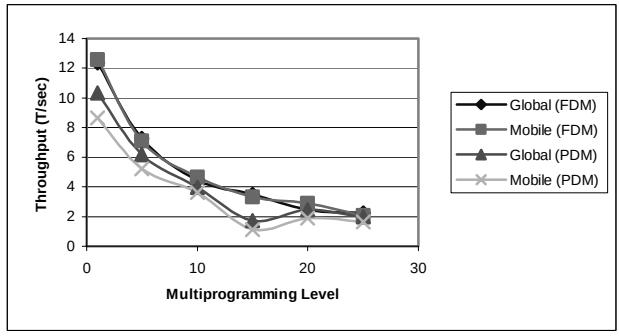


Fig. 4. Throughput with 50% Mobile Transactions

5.2 Future Directions

The proposed transaction processing system can be extended in a number of ways:

- Our simulation results showed the validity of the proposed transaction-processing model. However, it would be interesting to study the model in a real mobile computing environment A potential approach would be to implement the MDAS as part of the Mobile Computing Environment and simulation test bed (MCE) proposed in [16].

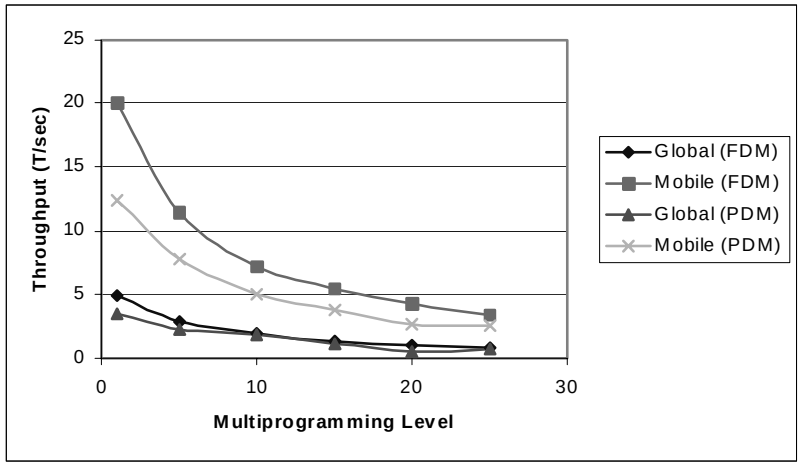


Fig. 5. Throughput with 80% Mobile Transactions

- The simulations were run with a fixed number of nodes in the system. The effect of varying the number of nodes on the system should be examined.
- The test results showed the performance of the system when either of the two operation modes was employed. The effect of mixed operation modes should be examined. It would be interesting to study the effect of mobility at the data sources level, as well.

As the final notes, the development of effective E-commerce technologies is in its formative stage. As E-commerce moves from a largely business-to-business model to include a proliferation of retail seeking channels, the demand for mobile data access will proliferate. The problems of effective mobile data access must be resolved to allow the effective development of electronic markets.

References

1. R. Alonso and H. F. Korth. Database System Issues in Nomadic Computing. ACM SIGMOD Conference on Management of Data, pp. 388-392, 1993.
2. B. R. Badrinath, et al. Structuring Distributed Algorithms for Mobile Hosts. Conference on Distributed Computing Systems, pp. 21-38, 1994.
3. D. Bell and J. Grimson. Distributed Database Systems. Addison-Wesley Publishing Company, 1992.
4. P. Bhagwat and C. E. Perkins. A Mobile Networking System Based on Internet Protocol. symposium on Mobile and Location Independent Computing, 1993.
5. Y. Breitbart, H. Garcia-Molina and A. Silberschatz. Overview of Multidatabase Transaction Management. VLDB 1(2): 181-239, 1992.
6. M. W. Bright, A. R. Hurson and S. Pakzad. Automated Resolution of Semantic Heterogeneity in Multidatabases. ACM TODS, 19(2): 212-253, 1994.
7. P. Chrysanthis. Transaction Processing in Mobile Computing Environment. Workshop on Advances in Parallel and Distributed Systems, pp. 77-82, 1993.
8. R. A. Dirckze and L. Gruenwald. Nomadic Transaction Management. IEEE Potentials, 17(2): 31-33, 1998.

9. M. H. Durham, A. Helal and S. Balakrishnan. A Mobile Transaction Model that Captures both the Data and Movement Behavior. Mobile Network Applications 2 (2): 149-162, 1997.
10. G. H. Forman and J. Zahorjan. The Challenges of Mobile Computing. IEEE Computer Volume: 27(4): 38-47, 1994.
11. J. Ioannidis, D. Duchamp, and G. O. Maguire. IP-based Protocols for Mobile Internetworking. ACM SIGCOMM Symposium, pp. 235-245, 1991.
12. L. Lamport. Time, clocks and the Ordering of Events in Distributed Systems. Communications of the ACM, 21: 558-565, 1978.
13. J. B. Lim, A. R. Hurson and K. M. Kavi. Concurrent Data Access in Mobile Heterogeneous Systems. Hawaii Conference on System Sciences, 1999.
14. S. K. Madria and B. Bhargava. A Transaction Model for Mobile Computing. Database and Engineering Applications Symposium, pp. 92-102, 1998.
15. K. Norvag, O. Sandsta and K. Bratbergsengen. Concurrency Control in Distributed Object-Oriented Database Systems. Proceedings of ADBIS, 1997.
16. R. Rajagopalan, S. Alagar and S. Venkatesan. MCE: An Integrated Mobile Computing Environment and Simulation Test bed. USENIX symposium on Mobile and Location Independent Computing, 1995.
17. A. P. Sheth and J. Larson. Federated Database Systems for Managing Dist. Heterogeneous & Autonomous Databases. Computer, 22(3): 183-236, 1990.
18. M. Wu and C. Lee. On Concurrency Control in Multidatabase Systems. Computer Software and Applications Conference, pp. 386-391, 1994.
19. X. Ye and J. A. Keane. A Distributed Transaction Management Scheme for Multidatabase Systems. International Trends in Electronics, 1: 397-401, 1994.
20. L. H. Yeo and A. Zaslavsky. Submission of Transaction from Mobile Workstations in Cooperative Multidatabase Processing Environment. Conference on Distributed Computing Systems, pp. 372-379, 1994.
21. V. Zwass. Structure and Macro-Level Impacts of Electronic Commerce: From Technological Infrastructure to Electronic Marketplaces. Foundations of Information Systems: [http://www.mhhe.com/business/mis/zwass/ecpaper.html]