1996

A Survey of Mobile Transaction Models

Abdelsalam Helal

Santosh Balakrishnan

Margaret Dunham

Ramez Elmasri

Report Number:
96-003

A SURVEY OF MOBILE TRANSACTION MODELS

Abdelsalam Helal
Santosh Balakrishnan
Ramez Elmasri
Margaret Dunham

CSD TR-96-003
January 1996

A Survey of Mobile Transaction Models

Abdelsalam Helal[1] Santosh Balakrishnan[2] Margaret Dunham [3]
Ramez Elmasri[2]

February 7, 1996

1 Introduction

With the availiability of powerful portable computing systems, and with the tremendous growth in the wireless communication technology, mobile computing is being projected as the future growth area in both academia and industry. Mobile computing systems when available will allow a user to be in constant touch with the his or her office computing resources enhancing productivity by efficiently utilising the commuting and travel time. Growth of the Internet and the popularity and effectiveness of the World Wide Web is another factor which has spurred the interest in mobile computing systems. The availability of the World Wide Web on mobile computing systems is expected to open up a new class of applications which provide location sensitive applications.

In this paper we present a survey of mobile transactions models proposed in the literature. These transaction models are aimed at maximizing concurrency and maintaining data consistency in a failure prone and low bandwidth mobile environment. Our main thrust in this paper is in comparing the transaction models on parameters like scalability, additional infrastructure, communication costs, exetensions required for commercial databases etc. These parameters are indicative of the cost of executing a transaction under a particular model as well as the cost of deployment of a transaction model. Both issues are crucial to the success of any model in the practical world.

The major factors that differentiate mobile computing from conventional computing is the low bandwidth and the frequent disconnections. A mobile user will typically be connected to the fixed network through a cellular network link or a radio or infrared link. In all these cases, the bandwidth available is very low typically in the range of 10 Kb/s in case of cellular links and 2 Mb/s in the case of an infra red link. As the mobile user moves, the current link may get disconnected, and the user may have to acquire a new link to retain connection to the fixed network. The mobile user may also stray off the service area of wireless providers or lose the connection for extended periods of time. These factors along with the issues arising out of mobility itself, like how to uniquely identify a mobile system in the internetworked world, how to service location sensitive queries like "Where is the nearest restaurant" has thrown up quite a good number of challenges and opened up a new field of research.

The research effort in mobile computing has been mainly concentrated in the following major areas.

- Networking.
- Operating Systems.

[1]Department of Computer Science, Purdue University, West Lafayette, Indiana 47907. E-mail: helal@cs.purdue.edu
[2]Department of Computer Science and Engineering, The University of Texas at Arlington, Arlington, Texas 76019. E-mail: {santosh,elmasri}@cse.uta.edu
[3]Department of Computer Science and Engineering, Southern Methodist University, Dallas, Texas 75275. E-mail: mhd@seas.smu.edu

- Database.
- System Architectures.

The mobile networking research has mainly concentrated in developing protocols for seamless access of a mobile computer in the internetworked world. The Mobile-IP Protocol [4] and other IP based internetworking protocols [17, 19, 26] attempt to modify the Internet protocol for use in mobile environments. These protocols define mechanisms that allow a mobile computer to access resources in an internetwork irrespective of its current location as well as allows other systems to access the mobile computer with the same IP address irrespective of the point of attachment of the mobile system. Operating Systems research in mobility has mainly concentrated on file systems mechanisms [22] and event delivery mechanisms [3]. Architecture issues and protocol mechanisms have been addressed in [23, 10]. Database issues has been another focal area in mobile computing research. The database related issues which has been studied in mobile computing reseach are transactions [12, 7, 11, 9, 27, 21], lock management [18], data consistency [13].

The paper is organized as follows, in the next section we describe the reference model for mobile environments. In section 3, the characteristics and issues related to mobile transactions is presented. In section 4, the transaction models on which the various mobile transaction models are based is described. In section 5, the various transaction models are presented, and in section 6, a comparison of the models is presented. In section 7, we outline the issues still to resolved.

2 Reference Model

The generally accepted reference model for the mobile computing environment is depicted in Figure 1. The model has a set of hosts on a fixed network, some of which serve as Base Stations (or Mobile Support Stations, as some authors refer to them). Each base station services a number of mobile hosts (or mobile nodes) which are currently in its cell. Mobile hosts as well as the base station have wireless interfaces. The base station communicates with the mobile hosts in its cell by broadcasting. As a mobile host moves across a cell boundary, the base station that was serving it prior to its movement, "hands over" the mobile host to the base station in the cell that the mobile host enters. A handoff protocol defines the actions that occur during the handoff. The model proposes the use of 'Location Databases' that are used to locate mobile hosts. A discussion of location management can be found in [2].

Mobile hosts are expected to be laptops or palmtops which have batteries with a short life-span. Further, the wireless communication channel between the mobile host and the base station is expected to be constrained by bandwidth. So, protocols have to designed so that the load on the mobile host as well as the amount of communication between the mobile host and the base station are minimized. This poses a challenge.

Mobile hosts are expected to be frequently disconnected from the fixed network. This may happen either due to physical damage suffered by the mobile hosts or due to a voluntary disconnection by the mobile user (in order to conserve power) The user may later connect to the fixed network at a different location. The network layer protocol has to be capable of handling these disconnections and providing a transparent interface to the upper layers. The Mobile Internet Protocol being developed by IETF [4] will probably be the accepted one for the network layer.

To summarize briefly, the mobile computing environment poses many challenges to hardware, software and communications. Most of the issues have already been addressed and solutions have been proposed. These are still under investigation and no formal standards have been yet adopted.

3 Mobile Transactions: Definition, Characteristics and Issues

According to the classical definition a transaction is described by its ACID properties. It has been long recognized that the ACID properties are too restrictive for many applications which can be modelled as

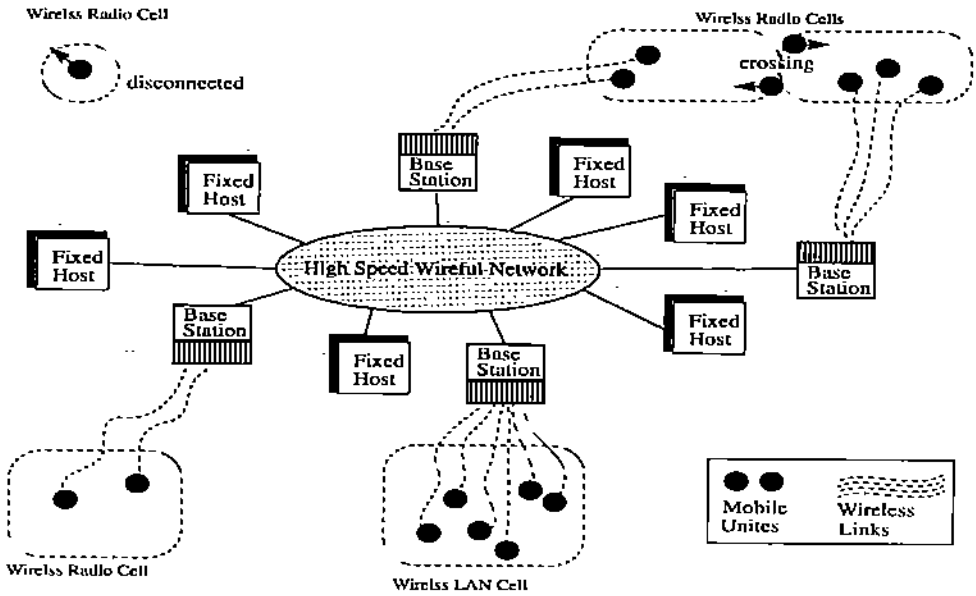

Figure 1: Architecture for Mobile Systems

transactions [20]. Many models have been proposed which extend the traditional transaction model by relaxing the atomicity, concurrency and isolation requirements [20]. The motivation for a new model for transactions in mobile environments can be best illustrated by the following examples.

Consider an application which allows a mobile user to order takeaway food from a fast food restuarant chain. The transaction is initiated by contacting the nearest restuarant in the chain address of which is obtained from a yellow pages server. As the order is being processed it is possible that mobile user is already past the service area of the first franchisee, and has entered the service area of the second franchisee. The execution of the transaction may now have to be relocated to the new serving franchisee of the mobile user. This process may continue till the order is completed and the user is ready to be served. The system may then direct the mobile user to the particular franchisee which will be serving the customer. This example illustrates an important characteristic of a mobile transaction, the ability to execute the transaction at a geographical proximity to the mobile user, which has issued the transaction.

Another example which illustrates the disconnected or asynchronous operation is adapted from [10]. Consider a mobile employee with a rather flexible travel plan. The employee may finalize his travel plans at some point on his trip and may initiate a transaction to book the most economical and convenient ticket to his destination. Once the transaction is initiated, the user may disconnect from the system. This sets in motion some agent processes, which determine the available choices. These agent processes report back the results when the user connects to the system again. Depending on the results the user may either decide to continue the transaction, or initiate a new transaction, or terminate the transaction. This example illustrates another important characteristic of mobile transactions, non-deterministic duratation and asynchronous operation.

These examples illustrate the need to relax the strict ACID properties of the traditional transactions, for mobile transactions. Transactions executed in mobile environments could be of non-deterministic duration, and could get aborted due to failures on the mobile unit end like power or connection failures. Thus even though transaction models have been designed for long duration transactions [20, 15], none of the existing transaction models is fully suited for operation in mobile environment.

3.1 Characteristics

From the above discussion the following characteristics of mobile transactions can be summarized.

Non Deterministic Lifetime. The mobile units from which a mobile transaction is issued will normally be attached to the fixed internetwork through a very low bandwidth wireless link. The routing delays in the mobile environment is far higher than in the fixed network. Mobile systems are prone to failures like battery power loss and wireless link loss. These factors coupled with other factors like disconnection from the fixed network either due lack of a communication link or for economic reasons makes the duration of a mobile transaction non deterministic.

Relocation. In order to maximize the response times and to efficiently use the limited bandwidth available, it is necessary to have the component of the mobile transaction executing on the fixed network to be as close to the mobile unit as possible. Thus it is also necessary to relocate the fixed network component of the mobile transaction as the mobile node moves.

3.2 Definition

A mobile transaction is a transaction of non deterministic lifetime submitted from a mobile capable node in a mobile heterogeneous environment. The mobile transaction can in general be considered to consist of two components, a mobile unit component and a fixed network component. The fixed network component of the mobile transaction may have to be partially or completely relocated as the mobile unit moves.

3.3 Issues

The mobile environment can be considered to be similar to highly distributed environments in many respects. But unlike in distributed environments, location of some hosts is not permanent in mobile environments. This along with the low communication bandwidth, frequent disconnections and high vulnerability throws up many challenges to researchers. In this section, we outline the issues addressed by researchers in mobile transaction design.

Data Consistency and Concurrency Control. In mobile environments, data could be replicated on a number of servers throughout the network. Some of these servers could be mobile units. Moreover, a mobile host might operate on cached data while being disconnected from the fixed network. The data conflicts arising in mobile environments could partly be due to the locality of the users accessing the data [16]. The execution of a mobile transaction could also be distributed and relocated amongst fixed hosts and the mobile nodes. The non-deterministic life times of a mobile transaction and the low bandwidth of communication links are other factors that affect concurrency control and cache management.

Infrastructure Requirements. For any model to be successful, it is important that it be moved from the research labs and deployed in the real world. Assuming a wireless communication infrastructure to be well in place, it is important to determine the additional resources required for having a mobile transaction system in place. These resources could range from protocols for location sensitive service access to mechanisms for optimized query management and controlled query release mechanisms.

Communication Costs. Bandwidth limitations and high costs of the communication links is one of the major constraints in mobile environments. Efficient utilization of bandwidth is thus very necessary in mobile environments.

Relocation Mechanisms and User Profiles. Mobile agents are processes or set of processes, that perform an activity on the fixed network on behalf of the mobile unit. These agents will typically be a transaction activity which access several databases, and report some results to the mobile node. Relocation of transaction execution or mobile agents is necessary to improve response times in mobile

environments. Effectiveness of mobile agent relocation has been studied in [23]. Performance can still be improved if the user profiles or user directives can be used to effect anticipatory relocation or to avoid unnecessary relocation.

Scalability. As mobile computing grows to be more affordable and popular, the number of mobile units handled by every base station could be large. Hence it is very important that a mobile transaction model scale up efficiently.

4 Applicable Transaction Models

The mobile transaction models presented in this paper are based on the extended transaction models developed for open ended long duratiuon transactions.

The extended transaction models which form the basis for mobile transaction models are

- Open Nested Transactions [20]
- Split Transactions [24]
- Saga - Compensating Transactions [15]

The applicability of these transaction models can be easily explained from the characteristics of mobile transactions. Due to the non deterministic lifetime of a mobile transaction, it is best caharacterized as a long lived transaction. The execution of a mobile transaction could be migrated or relocated as the mobile unit which issued the mobile transaction. The high vulnerability of mobile transactions may warrant arbitary rollback as well.

4.1 Open Nested Transactions

Open nested transactions are transaction models designed for long duration activities. These transactions typically consists of a set of sub-transactions which can be structured as a transaction tree. Open nested transaction model provide better support for long duration activities. These are also ideally suited for conversational transactions.

4.1.1 Properties of Open Nested Transactions.

Open nested transactions is a generalization multilevel transactions. In multilevel transaction, the subtransactions are divided into layers and the nesting depth is the same among all subtransactions. In the case of open nested transactions the restriction on nesting depth has been done away with, allowing different nesting depths in different subtransactions trees.

The classical ACID paradigm of transactions is too restrictive for long duration and other activities which can be easily modeled as transactions. Open nested transaction model accommodates these extended transaction activities by relaxing the ACID properties.

Atomicity. The classical definition requires transactions to be atomic at the lower details. Atomicity of a transaction or a subtransaction can still be enforced, if the system ensures that the effect or existence of an aborted transaction is hidden from other transactions and subtransactions. A completed open nested transaction cannot be rolled back by undoing the changes, since the results were already visible to other transactions. Open nested transactions are undone by executing compensating transactions which reverses the effect of the transaction.

Isolation. In the open nested transaction model, semantics of the transaction operations is used to relax the isolation of transactions. The operations are defined to be either commutative or compatible if the order of execution of the operations is insignificant for the success of the application. Result of an operation can be made available to its commuting or compatible operations, serializability is not compromised in this case, since the order of execution is immaterial in this case.

Durability In some long duration transaction applications, complete undo of a transaction may not always be acceptable. In open nested transaction model, programmers are allowed to tag sub-transactions with a *persistent* attribute. The updates of a *persistent* subtransaction is made persistent as soon as it completes. Compensating sub-transactions are not allowed to undo the effects of persistent sub-transactions.

4.2 Split Transactions

Split Transactions [24] or dynamically restructured transactions split a transaction into two independent serializable transactions. The transactions could be committed or aborted independent of each other. The operation *split-transaction* can be used to split a transaction into two. The split transactions are combined together by an inverse operation termed *Join Transaction*. Split transactions are mainly designed for user controlled open ended transactions.

4.2.1 Split Transaction Semantics

Split Transaction. The *split-transaction* operation is used to split a transaction into a set of independent entities. It takes the read and write sets as the input and produces a split.

split-transaction (Read(A), Write(A), Read(B), Write(B));

Join Transaction. The *join-transaction* is the inverse of the *split-operation*. It takes as input the target transaction to which the current transaction has to be joined. Let T be the current transaction to be joined with S. The operation

join-transaction(S)

joins T with S. All data items of T is now available to S. T may be committed or aborted depending on commit or abort of S. The join transaction can also be extended so that the join is done only if the target also agrees for the operation.

4.2.2 Properties of Split Transactions

Let a transaction T be split into two transactions A and B. Let the read and write sets of a transaction be denoted as Read(T) and Write(T) respectively.

Then

$$Read(T) = Read(A) \bigcup Read(B), and$$

$$Write(T) = Write(A) \bigcup Write(B).$$

The instructions in T are also split into instructions of A and B.

$$Instr(T) = Instr(A) \bigcup Instr(B).$$

The transaction is split if and only if the following conditions hold

$$Read(A) \bigcup Write(B) = \phi,$$

$$Write(A) \bigcup Write(B) = \phi, and$$

$$Read(B) \bigcup Write(A) = \phi.$$

From these properties it is evident that there are no dependencies between transactions A and B. They are entirely independent of each other. There are no serializability constraints between the transactions A and B, more over there are no data conflicts as well.

The above restriction can be relaxed, and if we assume that the transaction A is committed as soon as T is split, and only B is executed after that, then the properties can be written as

$$Write(A) \bigcap Write(B) \subseteq Writelast(B),$$

$$Read(A) \bigcap Write(B) = \phi, and$$

$$Read(B) \bigcap Write(A) = Share(A, B).$$

The set $WriteLast(T)$ is the set of data items written last by transaction T. $Share(A, B)$ is the data set shared by transactions A and B. The first property allows transaction B to overwrite the values written by A, but not vice-versa. The second property ensures that A precedes B. The third property allows transaction B to use values written by A.

4.3 Saga

Sagas [15] are defined as a set of relatively independent transactions. The transactions in a Saga are termed as component transactions. Each component transaction has a dual termed compensating transaction. A predefined order can be defined for the execution of the Saga. The transactions belonging to different Sagas can be interleaved in any fashion.

4.3.1 Properties of Sagas

As mentioned above a Saga consists of a set of component transactions and corresponding compensating transactions. A compensating transaction can semantically undo the effects of the component transaction. A component transaction can have ACID properties. A component transaction is not allowed to make any changes directly onto the database till it is ready to commit.

Let $T_1, T_2, ...T_n$ constitute a saga S. The saga S is said to commit if all the transactions $T_1, T_2, ...T_n$ belonging to S have executed and committed. If the saga has aborted after the commitment of the transaction T_k where $1 \leq k \leq n$ then the correct execution of the saga is

$$T_1, T_2, T_3, ...T_k, CT_{k-1}, ...CT_1$$

where CT_k is the compensating transaction for transaction T_k.

4.3.2 Limitations of Saga

The commitment of a saga is dependent on the commitment of all its components. But a saga by itself has no notion of commitment. This introduces a certain amount of inflexibility. More over some activities cannot to be modeled as Saga transactions, since they are inherently non-compensatable. The saga model has been extended to take care of these limitations.

4.3.3 Extensions of Saga Model

Vital and Non-vital components. Component transactions are distinguished as vital and non-vital components. A saga can commit if and only if all vital components commit. A saga need not be aborted if any of the non-vital components abort.

Nested Saga. A nested saga is a saga transaction which contains sagas as its components. A nested saga can be considered to be a set of non-vital components. Thus a nested saga can commit even if some of its components abort.

4.3.4 Non Compensating Transactions

Sagas are designed such that every component transaction will have a compensating transaction. But some transactions are inherently non-compensatable. Different techniques are used to accommodate this. In one method, non-compensating transactions are executed concurrently. In another technique non-compensating transactions are set up as nested transactions. In yet another technique additional semantics are used to specify dependencies between transactions.

5 Approaches to Mobile Transaction Models

In this section we describe the various approaches to mobile transaction modeling. The models we have considered have been proposed by Chrysanthis [7], Dunham and Helal [11], Pitoura and Bhargava [14], Walborn and Chrysanthis [21] and Nielsen [9]. All the models uses the mobile computing reference model described in section 2.

5.1 Reporting and Co-Transactions

This model [7] proposed by Chrysanthis is based on the Open Nested Transaction Model. A computation in mobile environment is considered to consist of a set of transactions, some of which may execute on the mobile node and some of which may execute on the fixed host. The model attempts to address the following two issues in particular

- Sharing of Partial Results while in execution.
- Maintaining computation state in a fixed node so that the communication cost is minimum.

The model allows

- Sharing of partial results.
- Transaction relocation.

The model proposes to modify Reporting and Co-Transactions [5, 6] to suit mobile environment. The model defines a mobile transaction to be a set of relatively independent transactions which interleave with other mobile transactions. A component transaction can be further decomposed into other component transactions allowing an arbitrary level of nesting.

Component transactions are allowed to commit or abort independently. If a transaction aborts, all components which have not committed yet may abort. Some of the transactions may have a compensating dual and may be compensated.

The model classifies Mobile transactions into four types. These are

Atomic Transactions Atomic transactions are normal components and may be compensatable with atomic compensating duals.

Compensatable Transaction These are atomic transactions whose effects cannot be undone at all. When ready to commit, the transaction delegates all operations to its parent. The parent has the responsibility to commit or abort the transaction later on.

Reporting Transactions Reporting transactions can make its results available to the parent at any point of its execution. It could be a compensating or a non-compensating transaction.

Co-Transactions Co-transactions behave in a manner similar to co-routine construct in programming languages. Co-transactions retain their current status across executions, hence they cannot be executed concurrently.

Properties of Reporting Transactions A reporting transaction reports its results to other transactions by delegating the results. A reporting transaction can have only one recipient at any given point of time. The changes made by a reporting transaction is made permanent only when the receiving transaction commits. If the receiving transaction aborts the reporting transaction aborts as well.

Properties of Co-transactions A co-transaction reports its results in a way similar to reporting transaction. But upon delegation the transaction stops execution and is resumed from the point left off. For any pair of co-transactions either both commit or both abort.

5.2 The Kangaroo Transaction Model

This model [11] is based on the global transactions and the split transaction models. In this model transaction relocation is achieved by splitting the transaction. A mobile transaction is considered as a global transaction in a Multi Database environment.

5.2.1 Reference Model

The mobile computing environment assumed has a small enhancement compared to the model described in the previous section. It consists of three layers, the innermost layer is the DBMS running on the source system. The outermost layer has the mobile nodes which initiate mobile transactions. The middle layer consists of a Data Access Agent (DAA). The Data Access Agent acts as a gateway between mobile nodes and the source system. The DAA is assumed to be present in every base system and acts as a router for the data. In general, the DAA is a Transaction Manager for mobile transactions.

5.2.2 Transaction Model

In the transaction model, the mobile transaction is termed as a *Kangaroo Transaction*. A Kangaroo transaction is a global transaction which is identified with the user who has issued it by a unique ID. A Kangaroo transaction (KT) consists of a set of *Joey Transactions*. A JT is associated with the base station or the cell in which it executes. When the mobile unit moves to a new cell, the JT in the previous cell is split, and one of the JTs is moved to the current cell of the mobile unit. Each JT may consist of a set of local and global transaction. The model is built upon the existing databases. The transactions are micro-managed by the individual database transaction managers.

5.2.3 Properties

Joey Transactions (JT) A Joey Transaction consists of a set of global and local transactions. Each JT should terminate in an abort, commit or split. A handoff of a mobile node from one cell to another will result in a split of the JT associated with it, if any. Each JT is identified by a unique ID assigned to it on its creation.

Kangaroo Transactions (KT) A Kangaroo Transaction consists of a set of Joey Transactions. For a KT to be successful, the last Joey Transaction in the order should end in a commit or abort, where as all other Joey Transactions should be split. The KT captures the movement behavior of the transaction.

5.3 The Clustering Model

This model [14] assumes a fully distributed system, and the transaction model is designed to maintain consistency of the database. The database is divided into clusters. A cluster defines a set of mutually consistent data. Bounded inconsistencies are allowed to exist between clusters. These inconsistencies are finally reconciled by merging the clusters. The model is based on the Open Nested Transaction model, extended for mobile computing. A transaction submitted from a mobile host is composed of a set of weak and strict transactions. Transaction proxies are used to mirror the transactions on individual machines as they are relocated from one machine to another.

5.3.1 Clusters

A cluster is defined as a "unit of consistency in that all data items inside a cluster are required to be fully consistent, while data items residing in different clusters may exhibit bounded inconsistency." Clusters can be defined either statically or dynamically. A wide set of parameters can be used for defining clusters. This could include the physical location of data, data semantics, and user definitions.

Consistency between clusters can be defined by an m-degree relation, and the clusters are said to be m-degree consistent. The m-degree relation can be used to define the amount of deviation allowed between clusters.

5.3.2 Weak and Strict Transactions

A mobile transaction is decomposed into a set of weak and strict transactions. The decomposition is done based on the consistency requirement. The read and write operations are also classified as weak and strict. The weak operations are allowed to access only data elements belonging to the same cluster, where as strict operations are allowed database wide access. For every data item, two copies can be maintained - one of them strict and the other weak. As mentioned above, a weak operation can access only the local copies of a data item. Weak operations are initially committed in their local clusters. They are once again committed when the clusters are finally merged. The weakly committed values are available only to other weak transactions belonging to the same cluster and not outside it.

5.3.3 Transaction Migration and Proxying

Transaction migration is used for relocating transactions to avoid long network delays, as well as to represent the user mobility. Relocation is denoted by T_{i-j}, which indicates that transaction T was partially executed partially at site i before it was migrated to site j. The Transaction proxy is modeled as a subtransaction, and includes the updates of the original transaction. The proxy is relocated, as the transaction/host moves. This is mainly for the purpose of recovery.

5.4 Semantic-based Mobile Transaction Processing

The semantics based mobile transaction processing scheme [21] views mobile transactions as a concurrency and cache coherency problem. It introduces the concepts of *fragmentable and* reorderable objects to maximize concurrency and cache efficiency exploiting semantics of operations defined on the data objects. The model assumes a mobile transaction to be a long lived one characterised by long network delays and unpredictable disconnections.

5.4.1 Exploiting Semantics for Concurrency and Caching.

Traditional definitions of concurrency and serializability is too strict for most operations [20]. Semantics of operations defined on an object can be utilised to define correctness criteria so as to maximize the concurrent operations on the object [8]. Both application dependent and application independent semantics can be utilised for this purpose.

Commutativity of operations is an important property which allow concurrent operations on an object. If certain operations on an object is commutative, then the database server can schedule these operations in an arbitary manner. Recovery also becomes quite simplified. Operations may be commutative either for all states or only for some states of the objects. The I/O values of the operations can be used to redefine serial dependencies of the operations. Though this may improve concurrency, it may require complex recovery mechanisms than the normal schemes. Organization of the object can be used for selective caching of the object fragments, necessary for continuing the operation during disconnected state. This approach reduces the pressure on the limited wireless bandwidth as well as utilises the cache space available on the mobile host better.

Application semantics can be utilised to define the *degree of inconsistency, degree of isolation* and the *degree of transaction autonomy* [20, 5]. Techniques like *epsilon serializability* and *quasi copies* [1, 25] can be used to specify allowable inconsistencies in the systems.

5.4.2 Fragmentable and Reorderable Objects

This approach utilizes the object organization to split large and complex objects into smaller easily manageable pieces. The semantic information is utilized to obtain better granularity in caching and concurrency. These fragements are cached and/or operated upon by the mobile hosts and later merged back to form a whole object. Thus the object fragments form the basic unit of consistency. A stationary server dishes out the fragments of an object on request from mobile units. The objects are fragmented by a split operation. The split is done using a selection criteria and a set of consistency conditions. The consistency conditions include the set of allowable operations on the object and the conditions of the possible state of the object. On completion of the transaction the mobile hosts return the fragments to the server. These fragments are put together again by the merge operation at the server. If the fragments can be recombined in any order then the objects are termed *reorderable* objects. Aggregate items, sets, and datastructures like stacks and queues are examples of fragmentable objects.

Formally an object O represented as $O = (S, C)$ where S is the state of the object and C is the set of consistency conditions is said to be fragmentable if it can be split into fragments (O_1,C_1), (O_2,C_2), .. (O_n,C_n) such that each of the fragments support the same set of operations as object O. The transactions can operate asynchronously on the object fragments. The modified objects when merged still satisfy the consistency constraints of the object O. The object O is reorderable if the fragments O_1, O_2 .. O_n can be merged in any order.

5.5 Time based Consistency Model.

In this model [9] a time based model is proposed to maintain consistency in a mobile environment. All objects in the system is associated with a set of time parameters. These time parameters are used to determine whether the object is currently consistent or not. A *modification time* (MT) is associated with every object on the server. When the object is cached on the mobile unit the modification time of the object on the mobile unit is set to the modification time of the object on the server. The object on the mobile unit is also associated with a *consistency time* (CT) and a *consistency flag*. These parameters are used to represent the consistency state of the object on the mobile unit.

5.5.1 Read Parameters.

When a mobile unit reads an object a parameter *Consistency Time Bound* (CTB) is associated with it. The object is considered to be consistent only for that time period. The CTB can be used to specify an optimistic or pessimistic approach. The model uses time locks for controlling access to objects as well. When a host acquires a read lock on an object a *Read Expiration Time* is set. The read lock is valid only for that duration.

5.5.2 Write Parameters

When a mobile unit writes a cached object a parameter *Modification Time Bound* (MTB) is set on the object. The mobile unit is required to update the server copy within the MTB failing which the modification is invalidated. When the server copy is updated the time parameters are appropriately updated on the server. Write locks are timed out after the *Write Expiration Time* (WET). It is necessary that a mobile unit establishes a connection within this period and update the copy on the server.

Time-out locks provide a flexible, secure mechanism to lock data objects. But this mechanism also requires that a mobile unit correctly estimate the time out period. Performance could be seriously affected if the time for which the lock is held is either too high or too low.

5.6 The Multidatabase Transaction Processing Manager(MDSTPM)

The MDSTPM is a model proposed be Yeo and Zaslavsky [27]. The model visualizes an environment where mobile hosts submit transactions to a coordinator on the fixed network. The mobile hosts may disconnect from the network. It might reconnect at a later time to query the result of the transaction. The system builds on existing heterogeneous, autonomous DBMSs and defines a new layer residing on top of them.

5.6.1 Architecture

The model assumes the MDSTPM to be running on top of each of the DBMSs. When a mobile host wants to connect to the fixed network, it does so by sending a message to a coordinator on the fixed network, requesting for a connection. The coordinator sends back an acknowledge message to the mobile host. Similarly, the mobile hosts send a disconnect request message to the coordinator when a voluntary disconnect is desired. These and other messages are handled asynchronously by the coordinator. The coordinator does not notice failures of mobile hosts until the failed host recovers and tries to reconnect. At that moment, the coordinator finds that the mobile host had disconnected abnormally by checking on a status table that it maintains.

The MDSTPM has the following components:

Global Communication Manager (GCM) This is responsible for handling message passing for the local site. It exchanges messages with mobile hosts as well as other sites on the fixed network.

Global Transaction Manager (GTM) This module manages the global transactions submitted to it from mobile hosts. The site to which the global transaction is submitted is designated the coordinator for that transaction. The other participants in that global transaction are termed Global Transaction Manager Participants (GTMPs). The Global Transaction Manager has components for scheduling global transactions (Global Scheduling Submanager) and for concurrency control of global subtransactions (Global Concurrency Submanager).

Global Recovery manager (GRM) This is responsible for recovery after a global transaction failure.

Global Interface Manager (GIM) This acts as the interface between the MDSTPM and the local DBMS.

5.6.2 Transaction Model

The global transaction submitted by the mobile host to the coordinator is scheduled and executed by the coordinator on behalf of the mobile host. When a transaction is submitted, it is put into an input queue by the GCM. The transaction undergoes a state transition when it moves from one queue to another. The queues (apart from the input queue) are the allocate queue, the active queue, the suspend queue and the output queue. The GSS schedules the execution of the transactions in the input queue and moves them to the allocate queue. The transaction gets the locks it needs from the GCS and moves to the active queue. The global transaction is broken down into subtransactions and dispatched to other sites by the GCM. When the global transaction has completed the first phase of the two-phase commit, it is put on a suspend queue. On completion, it is put on the output queue and handed over to the mobile host that initiated it, when that host connects to the network.

One of the significant features of this model is that once the mobile host has submitted the transaction to the coordinator, further communication is not required until the mobile host wants the results of the transaction. There is no mobility or migration of transactions.

6 Comparative Analysis of Transaction Models

In this section we present a comparative analysis of the transaction models presented above. We describe how each model address the issues mentioned in section 2.

6.1 Consistency & Concurrency

The issue of consistency and concurrency control has been addressed by the models in a widely varying manner. In the Reporting and Co-Transactions model [7] compensating transactions are used to maintain the consistency of data. Where as in Kangaroo Transaction Model [11] the underlying database is relied upon to maintain consistency. In the Clustered Data Model [14] the entire data model is designed around maintaining data consistency in a distributed environment. In order to improve concurrency the Reporting and Co-transaction model delegates/reports its operation to other transactions. Thus it makes its results available to other transactions.

Reporting and Co-Transactions In this model compensating transactions and delegation is used to maintain data consistency. In case of non-compensatable transactions only the local buffers are operated upon. Delegation transfers the responsibility of committing or aborting a transaction to the delegatee rather than on the transaction which conducted the operation. Delegation also allows the values to be used by other transaction.

Kangaroo Transactions The Kangaroo Transaction Model relies on the underlying transaction model to enforce data integrity. In this model the Transaction Manager splits a Kangaroo Transaction into a set of Joey Transactions, which are then executed on the underlying databases. A transaction is split when the Mobile node moves, and first transaction of the split gets committed immediately. Thus this releases some of the data items improving concurrency.

Clustered Data Model The design of Clustered Data Model revolves around maintaining data consistency in a fully distributed environment. The data in individual clusters are consistent and there can be bounded inconsistency between clusters. The transaction operations are classified depending on the type of data they access.

Semantics based Mobile Transaction Model The use of fragmentable and reorderable objects maximise concurrency as well as reduce the cost of caching. The mobile units operate on fragments of the data object, which is later pieced together again. All operations on the object fragments obey the consistency constraints specified in the consistency conditions specified when the fragment is dished out by the server.

Time Based Transaction Model This model proposes to use a time based locking to provide a secure and flexible mechanism. Performance of the system is highly dependent on duration for which a lock is held.

MDSTPM In this model, concurrency of global transactions is maintained by the Global Concurrency Submanager. The model assumes that the underlying DBMSs implement their own mechanisms for concurrency control and consistency of local transactions.

6.2 Additional Infrastructure Requirements And Compatibility with Commercial Databases

In this section we discuss the additional infrastructure assumed in each model. We assume that the reference model discussed in previous sections is available. The requirements discussed are apart from this. We also discuss whether the models can utilize the available databases to make available a database system for mobile users. All these models assume that the physical movement information is available to the upper layers.

Reporting and Co-Transactions This model does not assume any network interface other than the mobile environment discussed in the previous sections. The model adapts Reporting and Co-Transactions for mobile environment. Transaction relocation is also achieved using Reporting and Co-Transactions. The transaction manager will have to be modified to handle reporting and co-transactions. Since concepts like delegation and co-transactions is involved it will be difficult to implement this model as a wrapper around an existing database.

Kangaroo Transactions The kangaroo transaction model introduces the concept of Data Access Agents (DAA),which act as a router of transaction requests. The DAA is assumed to exist on a base station. Thus a base station will have to be enhanced to provide this facility. In this model a split is effected as soon as the mobile node hops from one cell to another, thus every cell should have a system capable of servicing a transaction. This model can be implemented over an existing database by implementing a Global Mobile Transaction Manager (GMTM). The GMTM can accept transaction requests from mobile capable node and assign it to individual database systems.

Clustered Data Model Though this model assumes no additional network facilities extensive database modification in required on the DBMS. New operations like Weak and Strict operations are defined. The data objects also have additional attributes reflecting their consistency requirements. Moreover other database operations like clustering of data and cluster merging also will have to be implemented.

Semantics based Mobile Transaction Model This model requires additional capability at both the server and the mobile unit end to split, operate and merge objects. Moreover the model also assumes that the database operates on objects so structured that fragmentation and merging is possible.

Time Based Transaction Model This model requires the database system to use time based locks. No more enhancements is proposed in the model.

MDSTPM This model is implemented by defining an MDSTPM layer over the existing DBMSs. This layer acts as an interface between the mobile hosts and the underlying Multidatabase system. The components of this interface include the GCM, GTM, GRM, and GIM. The queuing mechanism for transactions has to be implemented.

6.3 Communication Costs and Scalability

As mentioned previously communication costs form a significant factor in mobile transaction execution. The contributing factors to this is the cost of communication between the mobile node and the fixed server and the cost transaction relocation.

Reporting & Co-Transactions In this model communication between a mobile node and the fixed server takes place through co-transaction pairs or between a reporting transaction and co-transaction pairs. The results are communicated between the transaction. Apart from this housekeeping information will also have to be communicated between the nodes. These communication can form a bottle neck affecting scalability.

Kangaroo Transactions In the Kangaroo model once the transaction is fired from a mobile node only the final results and the housekeeping information need to be transmitted between the mobile node and the stationary server. The house keeping information will include log and the handoff information. The model requires a transaction to be split and relocated when its originating mobile node moves from one cell to another. This could be a costly operation since the data items to be committed have to be determined for each transaction split, moreover the transactions will have to relocated as well. These factors can affect scalability.

Table 1: Comparison of mobile transaction models.

	Consistency Concurrency	Database System Model	Additional Infrastructure
Reporting And Co-Transactions		Multi Database	None Required
Kangaroo Model	Relies on Underlying Database.	Heterogeneous Multi Database	Requires Data Access Assumes Extended S/W interfaces.
Clustering Model	Bounded Inter Cluster Consistency.	Fully Distributed Database.	None Required.
Semantics Based Model	Based on Object Semantics	Distributed Multi Database.	None Required.
MDSPTM	Relies on Underlying Database.	Heterogeneous Multi Database systems.	None Required.

Clustered Data Model As in the kangaroo model, the clustered data model require to communicate only the end results and the housekeeping information once the transaction is fired from the mobile node. As discussed previously the model divides the data into cluster and allows bounded inconsistency between clusters. Thus on every operation the cluster has to be maintained consistent and the inconsistency between the clusters also will have to be maintained within bounds. The cost of this increases as the size and number or clusters increases. Thus this is an important parameter determining the scalability of this model.

Semantics based Mobile Transaction Model Semantics based Mobile Transaction Model attempts to reduce the communication costs by caching only those parts of an object required in a disconnected operation. This reduces the pressure on the low bandwidth wireless network, as well as utilizes the mobile unit cache space better. As the number of transactions in the system increases the load on the server may increase correspondingly.

MDSTPM The communication costs of this model are significantly lower compared to the other models, since communication takes place only for submission and for transfering the results.

7 Open Issues in Mobile Transactions

In this section we discuss the issues what we feel is open for research in Mobile Transactions.

Network Transparency and Transaction Relocation All the network discussed in the previous section assumes that the physical movement information is available to the application layers. In the practical world one can safely assume that network layers like Mobile-IP [4] will be used. These network layer protocols attempt to provide a transparent disconnection free interface to the upper layers. Thus though the requirement for transaction relocation is very much understood, methods, cost and criteria of transaction relocation is not very much clear.

Programming Language Support and Location sensitive transaction operations Mobile Databases have to deal with the mobile computing issues like relocation of transactions, data inconsistencies, disconnections and low bandwidth links. Efficiency of the operations can be improved if these events are anticipated and responded to. SQL extensions can be proposed which will allow a developer to take care of these issues in the design itself. Semantics are also required to deal with the location sensitive queries that could arise in a database. New transaction operations may be required which allow manipulation of location information.

Table 2: Comparison of mobile transaction models.

	Net Management Comm. Cost	User Profile	Extensions Required For Commercial DB	Scalability
Reporting And Co Transaction	Hand Off Info Required. Reporting Co Transaction Info Exchanged.	Can be used for Relocating Transactions.	Transaction Manager will have to be extended to handle new transaction types.	Will Require High Bandwidth
Kangaroo Model	Hand Off info Required. Assumes that Each base station can handle transactions	Can be used to relocate transactions	Transaction Manager Should be able to handle split transactions and recovery mechanisms will have to be enhanced.	Splitting with frequent commits might load the database.
Clustering Model	Hand Off info Required.	Used to define clusters and for transaction migration	Transaction Manager should be enhanced to handle weak, strict transactions clusters definitions.	Large Number of clusters or large databases could lead to Cluster mgmt.
Semantics Based Model	Hand Off Info Required.	Not Required.	Objects should be Fragmentable or Reorderable. Object managers will be required.	
MDSTPM	No handoff info Required. Involves only Submission and querying of results	Can be used for priority queuing.	Requires the transaction Manager layer above the database system.	Transaction queuing could create a bottle neck.

Performance Evaluation of Mobile Transaction For a through comparison of various mobile transaction models its important that their performance be evaluated. There could be various measures of evaluation like response time, throughput, relocation costs, communication costs etc. The exact parameters and performance criteria for mobile transactions is not very clear and is very much an open issue.

8 Summary

Designing a transaction model for a mobile computing environment poses many challenges to researchers. In this paper, we introduced the issues involved in transaction processing in a mobile computing environment. We presented three advanced transaction models - Open Nested, Split and Saga - that have been adapted in mobile transaction models. We presented a comparative analysis of four mobile transaction models - Reporting and Co-transactions, Clustered data, Kangaroo transactions and the multidatabase transaction processing manager. We also enumerated some topics that are open for research.

References

[1] Garcia-Molina H. Alonso R., Barbara D. Data caching issues in an informational retrieval system. *ACM Transactions on Database Systems*, 15(3):359–384, Sept 1990.

[2] B. R. Badrinath, T. Imielinski, and A. Virmani. Locating strategies for personal communication networks. In *Proceedings of IEEE Globecom 92 Workshop on Networking for Personal Communications Applications*. IEEE, December 1992.

[3] Girish Welling Badrinath B. R. Event delivery abstractions for mobile computing. Technical Report LCSR-TR-242, Rutgers University, 1995.

[4] Editor Charles Perkins. draft-ietf-mobileip-protocol-13.txt. Technical report, IETF, 1995.

[5] Panos Chrysanthis. *ACTA, A Framework for Modeling and Reasoning about Extended Transactions.* PhD thesis, University of Massachusetts, Amherst, 1991.

[6] Panos Chrysanthis and K. Ramamritham. Synthesis of extended transaction models using acta. Technical report, University of Pitsburgh, 1993.

[7] Panos Kyros Chrysanthis. Transaction processing in mobile computing environment. In *Proceedings of IEEE workshop on Advances in Parallel and distributed Systems*, pages 77–82, Oct 1993.

[8] Ramamritham K. Chrysanthis P. K., Raghuram S. Extracting concurrency form objects: A methodology. In *Proceedings of ACM SIGMOD Conf*, pages 108–117, May 1991.

[9] Nielsen J. D. *Transactions in Mobile Computing.* PhD thesis, DIKU, 1995.

[10] David Chess, Benjamin Grosof, Colin Harrison, David Levine, Colin Parris, Gene Tsudik. Itinerant agents for mobile computing. *IEEE Personal Communications*, October 1995.

[11] Margeret Dunham and Abdelsalam Helal. A mobile transaction model that captures both the data and movement behaviour. 1996. Submitted to 12th International Conference in Data Engg.

[12] Pitoura Evaggelia and Bhargava Bharat. Revising transaction concepts for mobile computing. In *First IEEE Workshop on Mobile Computing Systems and Applications*, pages 164–168, Dec 1994.

[13] Pitoura Evaggelia and Bhargava Bharat. Consistent and recoverable agent-based access to mobile heterogenous databases. Technical report, Purdue University, 1995.

[14] Pitoura Evaggelia and Bhargava Bharat. Maintaining consistency of data in mobile distributed environments. In *Proceedings of 15th International Conference on Distributed Computing Systems.*, 1995.

[15] H. Garcia-Molina and K. Salem. Sagas. In *ACM Conference of Management of Data*, pages 249 – 259, May 1987.

[16] T. Imielinski and B. R. Badrinath. Data management for mobile computing. *SIGMOD Record*, 22(1):34–39, March 1993.

[17] John Ioannidis, Dan Duchamp, and Gerhald Q. Maguire Jr. IP-based protocols for mobile internetworking. In *Proceedings of SIGCOMM'91*, pages 235–245. ACM, September 1991.

[18] Ahmed Elmagarmid Jin Jing, Omran Bukhres. Dirstributed lock management for mobile transactions. *IEEE*, pages 118–125, 1995.

[19] David B. Johnson. Mobile host internetworking using ip loose source routing. Technical Report CMU–CS–93–128, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, Feb 1993.

[20] Elmagarmid Ahmed K. *Database Transaction Models For Advanced Applications.* Morgan Kaufman, 1991.

[21] Walborn Gary D. Chrysanthis Panos K. Supporting semantics-based transaction processing in mobile database applications. In *Proceedings of the 14th IEEE Symposium on Reliable Distributed Systems*, Sept 1995.

[22] James J. Kistler and M. Satyanarayanan. Disconnected operation in the Coda file system. In *Proceedings of the Thirteenth ACM Symposium on Operating Systems Principles*, pages 213–225, 1991.

[23] Liu George Y., Marlevi Alexander, Maguire Jr. Gerald Q. A Mobile Virtual-Distributed System Architecture for Supporting Wireless Mobile Computing and Communications. *Wireless Networks*, 2(1), 1996.

[24] C. Pu, G. Kaiser, and N. Hutchinson. Split-tnsactions for open-ended activities. In *Proceedings of the 14th VLDB Conference*, 1988.

[25] Leff A. Pu C. Replica control in distributed systems: An asynchronous approach. In *Proceedings of the ACM SIGMOD Conf*, pages 377–386, May 1991.

[26] F. Teraoka and M. Tokoro. Host migration transparency in IP networks: The VIP approach. *Computer Communication Review*, 23(1):45–65.

[27] L.H. Yeo A. Zaslavsky. Submission of transactions from mobile workstations in a cooperative multi-database processing environment. In *The 14th International Conference on Distributed Computing Systems*, pages 372–379, Jun 1994.