PERFORMANCE EVALUATION OF TRANSACTION PROCESSING IN MOBILE DATA BASE SYSTEMS

Ahmad Al-Qerem

Department of Computer Science, Zarqa University, Zarqa, Jordan

ABSTRACT

Specific characteristics of mobile environments make traditional transaction management techniques no longer appropriate. This is due the fact that the ACID properties of transactions are not simply followed, in particular the consistency property. Thus, transaction management models adopting weaker form of consistency are needed and these models can now tolerate a limited amount of consistency. In this paper we evaluate our execution framework on different mobile computation modes using different execution strategies. Moreover, the effects of the fixed host transaction are identified and included in the evaluation The integration between wired and wireless environments confirms that the execution strategy is critical for the performance of a system. Neither MHS nor FHS are optimal in all situations and the performance penalties and wasted wireless resources can be substantial. A combined strategy CHS over MCA-FSA computation model matches the best performance of the FHS and MHS and shows better performance than both in many cases.

KEYWORDS

Mobile Data Base, transaction processing, Execution Strategies, Execution framework, Performance evaluation.

1. INTRODUCTION

Mobile computing suggest that there will be more competition for shared data since it provides users with ability to access information and services through wireless connections that can be retained even while the user is moving. Further, mobile users will have to share their data with others. Those users may have access to the shared data by reliable wired communication (i.e. fixed host transaction or unreliable wireless communication (i.e. mobile host transaction). The task of ensuring consistency of shared data becomes more difficult in mobile computing because of limitations of wireless communication and restrictions imposed due to mobility and portability [20]. The access to the future information systems through mobile computers will be performed with the help of mobile transactions. Research in mobile database systems (MDBS) has received a lot of interests in the past decade [1, 16, 19, 24, 25, 26, 28, and 29]. Transactions in a mobile database system are usually associated with relaxation of ACID properties because of its long-lived nature and the limitation inherited from the wireless environment. Any abortion of a mobile transaction may result in a high cost associated with scarce wireless resources, while fixed transactions might only cause a little degraded level of system performance. In the past two decades, researchers have proposed various transaction models [13, 23,21] either to tackle the isolation and atomicity properties of mobile transaction [13, 23, 27] or guarantee the consistency of mobile transaction Most of the previous studies in mobile database system often assume that the system may consist of only one single type of transaction(i.e. mobile transaction). Without paying any attention to the effect of interaction between both types of transactions being shared the same databases simultaneously. Different assumptions have been made on the system and transaction models, e.g., the execution strategy and structures of the transaction, such that different scheduling techniques can be engineered to satisfy the different requirements of mobile transactions. However, little work has been done in the development of an integrated approach that can handle a MDBS satisfactorily where a mixed population of fixed and mobile transactions exists simultaneously. However, for many MDBS applications, such a mixture of workloads is very common [31]. It is very important that execution strategies designed for mobile transactions are evaluated in such mixture. In this paper an execution frame work is propose we investigate the effects of such interaction on the mobile resources under different execution strategies. In section 2 we show the difference between transaction in mobile environment and traditional transaction , section 3 describe the mobile database architecture, section 4 compare between three execution strategies, section 5 present the execution frame work used to compare the strategies, section 5 describe the simulation model, system's parameter and discuss the result, section 6 conclude our work.

2. MOBILE TRANSACTION PROCESSING

A transaction in mobile environment is different than the transactions in the centralized or distributed databases in the following ways.

1. The mobile transactions might have to split their computations into sets of operations, some of which execute on mobile host while others on stationary host. A mobile transaction shares their states and partial results with other transactions due to disconnection and mobility.
2. The mobile transactions require computations and communications to be supported by stationary hosts.
3. When the mobile user moves during the execution of a transaction, it continues its execution in the new cell. The partially executed transaction may be continued at the fixed local host according to the instruction given by the mobile user. Different mechanisms are required if the user wants to continue its transaction at a new destination.
4. As the mobile hosts move from one cell to another, the states of transaction, states of accessed data objects, and the location information also move.
5. The mobile transactions are long-lived transactions due to the mobility of both the data and users, and due to the frequent disconnections.
6. The mobile transactions should support and handle concurrency, recovery, disconnection and mutual consistency of the replicated data objects.

To support mobile transactions, the transaction processing models should accommodate the limitations of mobile computing, such as unreliable communication, limited battery life, low bandwidth communication, and reduced storage capacity. Mobile computations should minimize aborts due to disconnection. Operations on shared data must ensure correctness of transactions executed on both stationary and mobile hosts. The blocking of a transaction's executions on either the stationary or mobile hosts must be minimized to reduce communication cost and to increase concurrency. Proper support for mobile transactions must provide for local autonomy to allow transactions to be processed and committed on the mobile host despite temporary disconnection. Semantic based transaction processing models [1, 9] have been extended for mobile computing in [10] to increase concurrency by exploiting commutative operations. These techniques require caching large portion of the database or maintain multiple copies of many data items. In [10], fragmentability of data objects has been used to facilitate semantic based transaction processing in mobile databases. The scheme fragments data objects. Each fragmented data object has to be cached independently and manipulated synchronously. That is, on request, a fragment of data object is dispatched to the MU. On completion of the transaction, the mobile hosts return the fragments to the BS. Fragments are then integrated in the object in any order and such objects are termed as reorderable objects. This scheme works only in the situations where the data objects can be fragmented like sets, stacks and queues. In [11], the concept of transaction proxies is introduced to support recovery. For each transaction submitted at an MU, a dual transaction called proxy is submitted to the base station. The proxy transaction includes the updates of the original transaction. Proxy transaction takes the periodic backup of the computation performed at mobile host. Similar to above, in [12], the notion of twin-transaction was introduced which essentially replicate the process of executing transactions. In twin transaction model, each writ's request will be mirrored and two equivalent transactions will be created. In this way, if a mobile host is disconnected, the transaction execution can still be proceed. In [11], disconnection was not discussed. Dynamic object clustering has been proposed in mobile computing in [8, 13]. It assumes a fully distributed system, and the transaction model is designed to maintain the consistency of the database. The model uses weak read, weak-write, strict-read and strict-write. The decomposition of operations is done based on the consistency requirement. Strict-read and strict-write have the same semantics as normal read and write operations invoked by transactions satisfying ACID properties. A weak-read returns the value of a locally cached object written by a strict-write or a weak-write. A weak-write operation only updates a locally cached object, which might become permanent on cluster merging if the weak-write does not conflict with any strict-read or strict-write operation. The weak transactions use local and global commits. The local commit is same as pre-commit of [6] and global commit is same as final commit of [14]. However, a weak transaction after local commit can abort and is compensated. In [14], a pre-committed transaction does not abort; hence require no undo or compensation. A weak transaction's updates are visible to other weak transactions whereas prewrites are visible to all transactions. [4] Present a new transaction model using isolation-only transactions (IOT). The model supports a variety of mechanisms for automatic conflict detection and resolution. IOTs are sequences of file accesses that unlike traditional transactions have only isolation property. Transaction execution is performed entirely on the client and no partial result is visible on the servers. IOTs do not provide failure atomicity, and only conditionally guarantee permanence. They are similar to weak transactions of [8]. An open nested transaction model has been proposed in [2] for modelling mobile transactions as a set of subtransactions. They introduce reporting and co-transactions. A reporting transaction can share its partial results, can execute concurrently and can commit independently. Co-transactions are like co-routines and are not executed concurrently. The model allows transactions to be executed on disconnection. It also supports unilateral commitment of subtransactions, compensating and non-compensatable transactions. The author claims that the model minimizes wired as well as wireless communication cost. However, not all the operations are compensated [2], and compensation is costly in mobile computing. Transaction models for mobile computing that perform updates at mobile computers have been developed in [15, 8]. These efforts propose a new correctness criterion [2] that is weaker than the serializability. They can cope more efficiently with the restrictions of mobile and wireless communications. In [14, 16.17], we look mobile transaction more as a concurrency control problem and provide database consistency. We incorporate a prewrite operation [5] before a write operation in a mobile transaction to improve data availability. A prewrite operation does not update the state of a data object but only makes visible the value that the data object will have after the commit of the transaction. Once a transaction received all the values read and declares all the prewrites, it can pre-commit at mobile host (i.e., computer connected to unreliable communication) and the remaining transaction's execution is shifted to the stationary host (i.e., computer connected to the reliable fixed network). Writes on database, after pre-commit, take time and resources at stationary host and are therefore, delayed. This reduces network traffic congestion. A precommitted transaction's prewrite values are made visible both at mobile and stationary hosts before the final commit of the transaction. This increases data availability during frequent disconnection common in mobile computing. Since the expensive part of the transaction's execution is shifted to the stationary host, it reduces the computing expenses (e.g., battery, low bandwidth, memory etc.) at mobile host. Since a pre-committed transaction does not abort, no undo recovery needs to be performed in our model. A mobile host can cache only prewrite values of the data objects, which will take less space, time, and energy and can be transmitted over low bandwidth. A kangaroo transaction (KT) model was given in [3]. It incorporates the property that transactions in a mobile computing hop from a base station to another as the mobile unity moves. The mobility of the transaction model is captured by the use of split transaction [18]. A split transaction divides on going transactions into serializable subtransactions. Earlier created subtransaction is committed and the second subtransaction continues its execution. The mobile transaction is split when a hop occurs. The model captures the data behaviour of the mobile transaction using global and local transactions. The model also relies on compensating transaction in case a transaction aborts. The model in [6]has the option of either using nested transactions or split transactions. However, the save point or split point of a transaction is explicitly defined by the use of pre-commit. This feature of the model allows the split point to occur in any of the cell. Unlike KT model, the earlier subtransaction after pre-commit can still continue its execution with the new subtransaction since their commit orders in the model [6]are based on pre-commit point. Unlike KT, the model in [6] does not need any compensatory transaction. In [7], a distributed lock management scheme is presented. It allows a read unlock for an item to be executed at any copy site of that item; the site may be different from the copy site on which read lock is set. The proposed scheme utilizes the replicated copies of data items to reduce the message cost incurred by the mobility of the transaction host. In a multidatabase environment with mobile computers involved, the nature of computing is such that the user may not wait for the submitted global transaction to complete before disconnected from the network. In [19], a basic architectural framework to support transaction management is multidatabase systems is proposed and discussed. A simple message and queuing facility is suggested which provides a common communication and data exchange protocol to effectively manage global transactions submitted by mobile workstations. The state of global transactions is modelled through the use of subqueues. The proposed strategy allows mobile workstations to submit global transactions and then disconnected itself from the network to perform some other tasks thereby increasing processing parallelism and independence. A transaction management model for the mobile multidatabase is presented in [7], called toggle transaction management technique. Here site transactions are allowed to commit independently and resources are released in timely manner. A toggle operation is used to minimize the ill-effects of the prolonged execution of long-lived transaction.

3. MOBILE DATABASE ARCHITECTURE

As discussed in previous section, there can be different models for mobile computing systems. Among them, we choose the model most likely to be a computing environment for mobile database systems. In the system, some computers are fixed and some are mobile. Among the mobile computers, some play the role of mobile clients and some the role of mobile hosts. This architecture is widely accepted in existing research for mobile database applications [2], where a global database is distributed among the fixed network nodes. An example of such architecture is shown Figure 1. In this system architecture, all the network nodes in the wired part are fixed units. Mobile units retain communication through the wireless links. Some fixed units, called base stations or mobile support stations, have special functionality with a wireless interface to communicate with mobile units. Each mobile station provides networking services for all the mobile units within a given geographic area called cell. In other words, each cell has a base station and mobile units within that cell access data on remote nodes through the local base station.

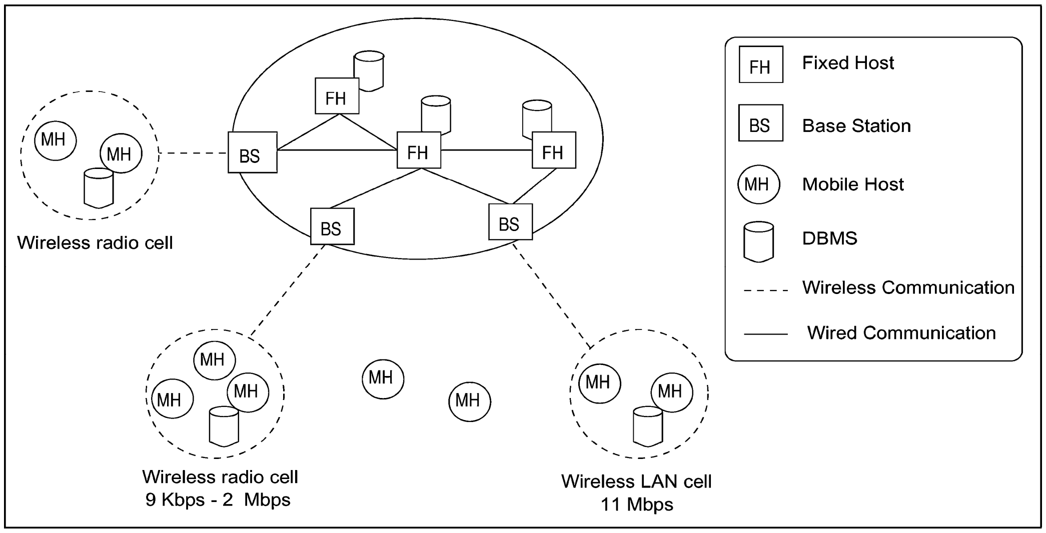


Figure 1. Architecture of mobile database systems

A global database (GDB) is defined as a finite set of data items. A GDB is partitioned among fixed hosts as well as mobile hosts (GDB = MDB $\cup$ FDB). Where the data items on fixed hosts make up the fixed database (FDB) of the GDB (i.e. FDB = $\cup FDB_i$) and the data items on mobile hosts make up the mobile database (MDB) of the GDB (i.e. MDB = $\cup MDB_i$). Moreover, the data items on a single mobile host are called a mobile part of the MDB. Accordingly, a data item in the FDB is referred to as a fixed data item and a data item in the MDB is referred to as a mobile data item. Furthermore, each data item has one primary copy and thus one owner. Only its owner can update the primary copy of a data item. A data item in the MDB is owned by a mobile host and a data item in the FDB is owned by a fixed host. We further assumed that each mobile host has local storage and computing capability, and is able to estimate and exchange its status regarding the mobility information. The fixed data items are replicated by caching on a mobile host, whereas the mobile data items have a full replication on a fixed host. Moreover, a mobile transaction does not access the data on other mobile units (MDBi $\cap$ MDBj = φ for all i, j such that i $\neq$ j). That is, it can access only local mobile data items and fixed data items.

4. Execution Strategies

4.1. Fixed Host Execution Strategy (FHS)

In this strategy the execution of mobile transactions with a copy of required data on the mobile unit is transferred to the fixed host. The fixed host coordinates the execution of the transaction on behalf of the mobile host and returns the final results back to the mobile unit. As such, a mobile transaction can be processed just like a traditional distributed transaction in a client server model. The only difference is that the propagation of updates and the return of results to a mobile host may be delayed as a consequence of wireless communication. This execution strategy is relatively simple. Its main advantages are:(1) maintaining strict data consistency because an entire transaction is managed and Executed on fixed host, traditional transaction schemes can be applied, (2) reducing battery consumption because the operations are shifted to the fixed network, computation tasks on a mobile host are avoided and (3) the level of concurrency at the fixed host is increased due to avoidance of long delays result form poor Communication. However, a shortcoming of fixed host execution strategy is that no autonomous operations are allowed during the disconnection of a mobile unit because required data on the fixed host is not available. This policy is suitable when the entire database is allocated on the fixed machines. In other words, a mobile host only plays the role of a client, or a remote device. This strategy has been taken an underlying assumption, of the processing model, by many researchers.

4.2. Mobile Host Execution Strategy (MHS)

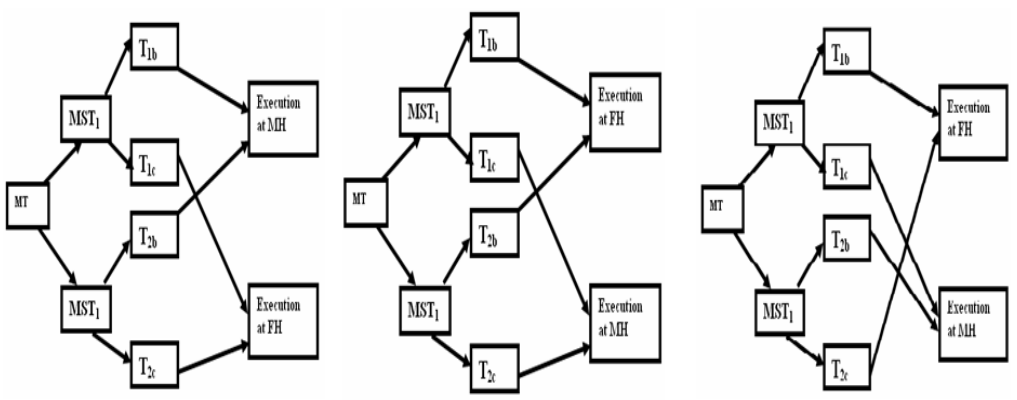
To allow continuous computation when a disconnection occurs, the data stored at the fixed hosts can be duplicated on a mobile unit rather than moving data to a fixed host. Instead of making a full replication on a local disk, using cached copies is a common technique to support autonomous operations, increase the availability [14], and minimize network access. Since data involved in a mobile transaction is always locally available, this policy allows transaction processing independent of fixed data services. Hence, the autonomous operations can be carried out at the mobile unit. In addition, this policy uses less battery power compared to the FHS strategy, regarding data transmission, because the network connection between a mobile unit and a fixed host is asymmetric. A fixed host typically has a stronger transmitter and unlimited power. Therefore, transporting data from a fixed host to a mobile unit is likely to be more efficient than transporting data in the opposite direction. There are two method of transporting data. Firstly, in planned mode. In this mode required data is transferred from fixed hosts to a mobile unit before a disconnection occurs. This mode requires that a disconnection protocol knows which data will be used in the near future. Secondly, non planned mode. In this mode, data is downloaded based on current transaction requirements and network conditions. For instance, if a transaction uses data items X and Y, and at this time point the bandwidth is high, then the data will be transferred to the mobile unit. Considering two types of disconnection, the first mode is more suitable for predictable disconnection, while the second mode is suitable for unpredictable disconnection.

4.3. Combined Execution Strategy (CHS)

For mobile transactions having a number of subtransactions which pass through different resources availability during their execution life time, neither MHS nor FHS on its own can give better performance in terms of system throughput and battery consumption in al1 situations. Since the conditions of the mobile environment are dynamically reflected in the transaction processing, an adaptive approach to control the execution of a transaction is desirable. Once that takes fixed host and mobile host execution strategies as two basic options for each subtransaction, a decision is made for a given transaction request based on available mobility information to select a specific execution strategy for each subtransaction of the same mobile transaction in an optimal way corresponding to the current mobile computing environment. With the combined strategy, a mobile subtransaction can be scheduled with a data request or a transaction request. In this approach we can gain two main advantages: (1) a mobile transaction can be tentatively committed based on locally cached data and other transactions can be submitted during disconnection so that autonomous operations can be supported. (2) A mobile transaction can be either executed on a mobile host or a fixed host based on a run-time decision.

5. EXECUTION FRAMEWORK

There have been many different models proposed for Mobile Transactions (MT) [2]. In these models mobile transaction supposed to be decomposed in to set of subtransaction which make it possible for the atomicity to be relaxed. The common base between these models their extension of advanced transaction models. Since our work here is concentrate on evaluating execution strategies under different network connectivity conditions rather than evaluating a specific mobile transaction model. We make a general assumption that a mobile transaction MT is defined as a set of mobile subtransactions. The update made by a subtransaction at the execution place should be reflected on the opposite part of the network. (I.e. if the execution takes place at the mobile host, the update made should be reflected on the fixed host and vice versa). So each mobile subtransaction SMT is decomposed into two subtransactions: namely, a basic subtransaction Tb and a complementary subtransaction Tc. Corresponding to this, there are two commit points: local commit and global commit, respectively. A local commit records the status of all basic subtransaction processing when their processing is done, as it may not be possible for their complementary sub-transactions to be issued due to network disconnection at this time. A global commit implies that all complementary subtransaction has been executed and modified data items have been propagated to their master copies. Global commit can happen only when the network is connected. Successful commitment of a mobile transaction occurs if and only if their basic and complementary sub transactions are successfully committed. So the processing of each mobile subtransaction consists of two phases, a basic subtransactions executes in the first phase of processing a mobile subtransaction. Major transaction operations are performed in this phase. A complementary subtransaction derived from a basic subtransaction and occurs in the second phase of mobile subtransaction processing and its execution takes place when the data items on both the fixed part and mobile part are connected. A complementary subtransaction largely performs updates on the data items that are modified by its basic subtransaction and the place where a basic subtransaction executes is always opposite to the one where its complementary subtransaction does. In general each subtransaction of mobile transactions is scheduled as a data request or a transaction request. The basic subtransaction of a data request transaction is processed on a mobile transaction host with cached data from the fixed part. The corresponding complementary subtransaction is processed on a fixed host. Conversely, the basic subtransaction of a transaction request is processed on a fixed host with the replication of the data items from the mobile host. The matching complementary subtransaction is processed on the mobile host. In this way, the effect of data modification can be propagated to its master copy. When a mobile subtransaction is issued from a mobile host, its basic subtransaction is processed with cached data if network connectivity between the mobile transaction host and its local base station is unsatisfactory (i.e., poor bandwidth or disconnected). If the required data is not available, the transaction is aborted, otherwise after completed; a complementary subtransaction is issued and put into a queue. When the mobile transaction host restores its connection, the queued complementary sub transactions are first submitted through a base station to a fixed host. If a complementary subtransaction fails, the mobile subtransaction is aborted: otherwise it should be waiting for global commitment. If network connectivity is high a mobile transaction is started. The location where its basic subtransaction is processed depends on the execution strategy see Figure 2(a-c)



(a) MHS strategy (b) FHS strategy (c) CHS strategy

Figure.2 Execution Framework for the three strategies

If the decision is made to use a data request, the fixed data items required by the transaction are downloaded from the fixed host (the size of downloaded data is determined by the amount of valid data in the cache of the mobile transaction host). After this, the process of mobile transaction processing is similar to the one above when network connectivity is poor. On the other hand, if the decision is made to use a transaction request. The mobile data items required by the transaction are uploaded from a mobile part owned by the mobile transaction host, to a fixed host, where the basic subtransaction is processed. Similarly, at this time, if network connectivity between the mobile transaction host and its local base station is below some threshold, the complementary subtransaction will be queued and executed later when network connectivity is improved. If network connectivity remains strong, the complementary subtransaction that performs updates of mobile data items on the mobile transaction host can be issued as soon as the basic subtransaction is completed. The execution of mobile transaction can be described as pseudo code in Figure 3.

```
Mobile transaction Processing
  Begin
For each mobile subtransaction MST
Mobile host analyzes the subtransaction MST to build $T_{basic}$ and $T_{complemnetary}$ based on its read and write requests and cash status (we assume that MH has some server capability).
  If the subtransaction T is scheduled as D-req transaction then
    Begin
For all data item reads and writes by $T_{basic}$, if $D_i$ not in the $MDB_i$ cash, MH sends a request to MTM for all the required read values and request for lock. After MTM acquires the necessary locks, it returns values to MH. Execute the basic transaction on the MH and send $T_{complementary}$ to be executed on the fixed host.
    End
 Else if the subtransaction T is scheduled as T-req transaction then
  Begin
For all data item reads and writes by $T_{basic} \in MDB_i$
 MH sends a request to MTM for all operation with data items $D_i \in MDB_i$ required by these operations. After MTM acquires the necessary read locks, Execute the basic transaction on the FH and send $T_{Complementary}$ to the mobile host.
End
If all mobile subtransactions successfully finish their execution then
 Mobile transaction is committed
  End
```

Figure.3 Mobile Transaction Processing Algorithm

6. COMPUTATION MODEL

The frame work adopted in this paper could be used in different mobile computing models. There is many computing model found in the literature; in [32] the author restrict the computation models to a five model and a comparative study based on the ability to deal with mobility dimension is proposed. Actually our framework can be utilized in the first three computation models explained as follows:

Mobile Client - Fixed Server Model (MC-FS Model): where the 'client process' requests for a service on a different computer, the server that services the client's request by performing computations, managing data and all other shared resources on a fixed network and this will be mange by the application located at the fixed network as shown in Figure 4.

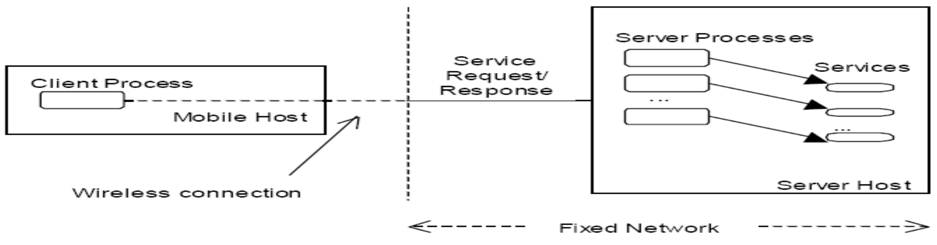


Figure.4 MC-FS Model

Mobile Client – Fixed Agent – Fixed Server Model (MC-FA-FS Model): This model the same as client server model with augmented agent on the fixed network to his delegator. Actually such model is proposed to overcome the connection's variability and disconnection problem. See fig 2

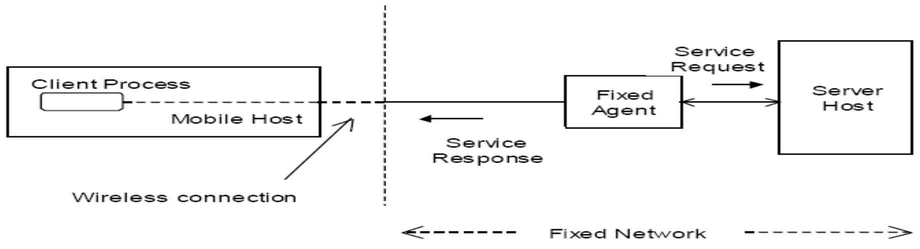


Figure.5 MC-FA-FS Model

Mobile Client with Agent – Fixed Server with Agent Model (MCA-FSA Model): where the agent at the client responsible for transaction and data management on behalf of mobile client. See figure 3

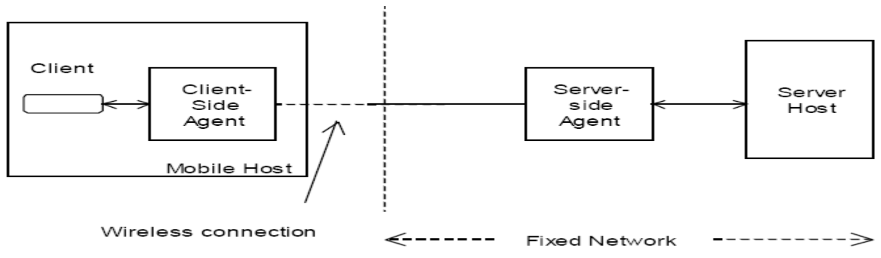


Figure.6 MCA-FSA Model

A comparative overview of these models and other models that is not appropriate to be used in our frameworks is given in [32].

7. PERFORMANCE EVALUATION

The experiments were designed to study two dimensions: the first was concerning in the impact of mobility and disconnection conditions on different execution strategies were defined by our framework ; where as the second dimension is to compare and contrast the different execution strategies using different computation models. The experiments were conducted first using the first computation models (i.e. MC-FS) in order to study the execution strategies themselves, and then we study the effect of those strategies on different computation models. In the former experiments we study the response time, power consumption and throughput. Where the later experiments we study the effect of the computation models namely MC-FS, MC-FA-FS and MCA-FSA average number of restart per transaction using different execution strategies to find out which computation models that fit with which execution strategy.

7.1. Simulation Mode1

The mobile system is abstracted as a queuing model. Its general structure is shown in Figure4. The model consists of three servers labeled MH, MTM, and FH. The server MTM stands for the mobile transaction manager related services. Normally, a mobile transaction request gets this service first. Each mobile transaction issues a set of requests and each request represents one of its subtransaction which could be a data request or a transaction request transaction based on the execution strategy being applied. The server MH represents the services provided by a mobile transaction host and the server FH the services provided by a fixed host where transaction coordination takes place. If the subtarnsaction of a mobile transaction is scheduled as a Dreq-transaction, when its basic subtransaction has finished its execution, the complementary subtransaction is put into queue MQ1. If the network is connected, a Dreq complementary subtransaction is submitted to a fixed host for processing and enters a queue FQ1; otherwise it has to wait in MQ1 for network re-connection. If a Dreq complementary subtransaction is successfully finish its execution. The mobile subtransaction is completed; otherwise, the subtransaction fails. This process will be repeated for all sub transactions, when all subtransactions are successfully completed, the entire mobile transaction is committed. For a subtransaction that schedule as Treq, this process is identical; but with different notations of Figure 4; that is if the network is connected, a T-req complementary subtransaction is submitted to a mobile host for processing; otherwise it has to wait in MQ2 for network re-connection. The CHS strategy can be viewed as a combination of the two basic strategies MHS and FHS. For the MHS approach, each subtransaction in a mobile transaction goes through four processing steps: data downloading, basic subtransaction processing, complementary subtransaction submission, and complementary subtransaction processing. Thus, there are four simple services. Similarly, the FHS approach has four simple services as well. The difference between the FHS model and MHS model is that the data required by a mobile transaction is uploaded from the mobile host to a fixed host. Besides, the first three simple services, i.e. data uploading, basic subtransaction processing and complementary subtransaction submission, are implemented on a fixed host. Whereas complementary subtransaction processing is performed by the mobile host that issues the transaction. According to the BHS strategy, when a mobile host is disconnected a transaction is treated as a Dreq transaction and has no data transmission. If the network is connected, the data will be uploaded or downloaded based on current scheduling decision. Generally, the simulation is conducted for the three strategies separately.

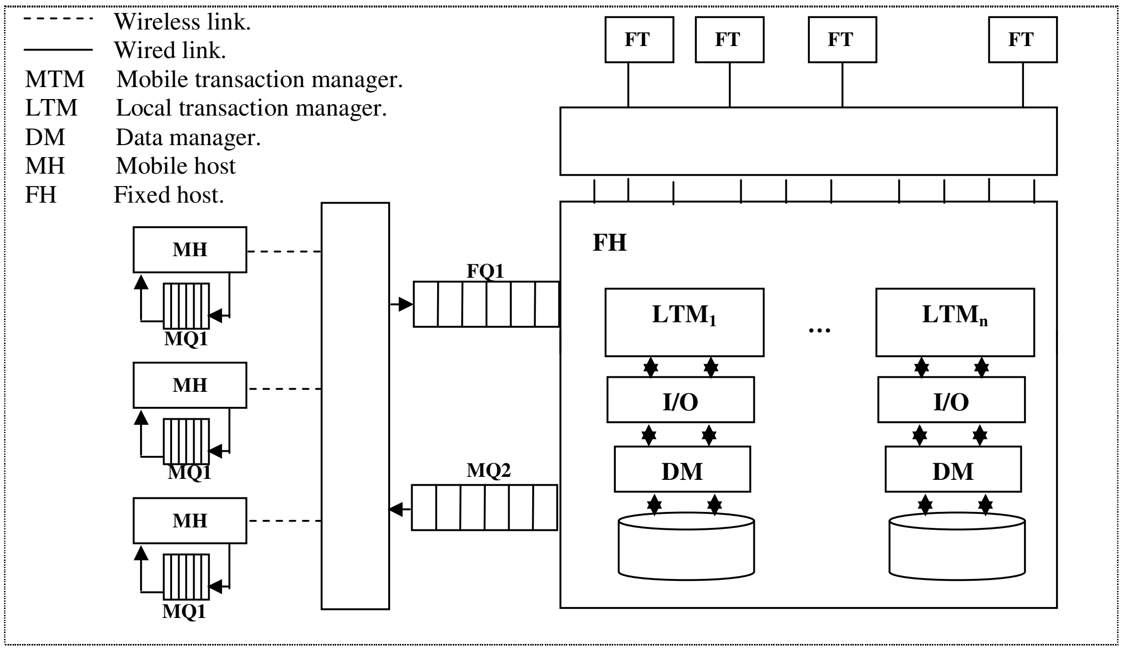


Figure.7 The simulation model overview

If the policy is FHS, transaction requests are always scheduled as Treq transactions. If the network is disconnected when a request arrives, this request must give up because its first phase cannot be carried out. After the completion of the first phase, if the network is not connected at this time, the complementary subtransaction cannot be submitted and must wait in the queue. Unlike the FHS approach, the MHS strategy usually does not discard a transaction request. Instead, the first phase of transaction processing is performed by using cached data even if the network is disconnected. However, if the network is disconnected after this phase, the request must be queued for later processing. For the BHS strategy, when the network is disconnected, its work likes an MHS approach; otherwise, it functions as a mixture of the FHS and MHS. Moreover, a complementary subtransaction may or may not be blocked. The network connectivity directly impacts the performance of transaction processing. To reflect this, the time spent on the completion of a Dreq transaction can be distinguished into four cases based on the network connectivity during transaction execution. (1) A CC-type transaction is just like a normal transaction where the network is in a fully connected condition. The other three types of transactions indicate that at least one of two communication steps is broken (denoted by the D). (2) CD-type transaction indicates that the network is connected at the time of the data transmission and disconnected at the time of complementary subtransaction submission. (3) DC-type transaction indicates that the network is disconnected at the time of the data transmission and connected at the time of complementary subtransaction submission. (4) DD-type transaction indicates that the network is disconnected at the time of the data transmission and the time of complementary subtransaction submission. The performance of these four types of transaction can be different. The functions to determine the execution and communication time is related to the current state of the environment. In general the service type in the system can be classified into two type communication oriented and processing oriented. For a communication oriented service. The service time consists of a constant communication overhead and the data transport time, which is determined by the size of data and the bandwidth at the time of data transfer. For a processing oriented service, the service time is basically determined by the number of accessed data items, the average number of operations on these items and the amount of time spent in the fixed host which is based on different factors specified in section 6.2.

7.2. System Parameters

The underlying mobile database system is composed of a number of databases distributed among fixed and mobile hosts. Transaction-generators are responsible for creation of transactions to be executed in the system. Parameters controlling the generation of transactions include the workload (the number of concurrent transaction per unit of time allowed in the system); the average, minimum, and maximum number of subtransactions in each mobile transaction; the average, minimum, and maximum number of database operations in a subtransaction; the probability of a write operation. The global workload consists of randomly generated local and mobile transactions spanning over a random number of sites. At each local site, there are a number of local transactions. The local system does not differentiate between the two types (local transaction and mobile subtransaction). Transactions enter the execution phase are subsequently scheduled by acquiring the necessary lock on their data items. If the lock is granted, the operations proceeds through the CPU and I/O queue, and for a mobile subtransaction an abort or commit signal is communicated back to the MTM and the subtransaction terminates. The local system may abort a local transaction or a mobile subtransaction at any time. If a mobile subtransaction is aborted locally, it's communicated to the MTM and the mobile transaction is aborted at all sites and restarted. The communication between the mobile host and the MTM can occur in both direction using either an uplink channel for uploading data or downlink channel for downloading. The communication overhead for uplink and downlink channels is 30 and 10 respectively. When the disconnection occurs, the mobile unit is disconnected for 20-30 second. In order to effectively evaluate the different execution strategies, several parameter are varied for different simulation runs. Most of these parameters are given in Tables 1, 2 and 3 along with their default values.

Table 1. Communication Parameters

Communication Parameters	**Default Values**
Arrival of mobile transaction	5
Arrival of fixed transaction	5
Mobility timer	5,10,15,..., 50
Disconnection Threshold	1000 , 500, 100
UplinkBW	10-1500
DlinkBW	10-3000
Disconnection Period	20 – 30 Second
Service time for each communicated message using Uplink Channel	30
Service time for each communicated message using Downlink	10

Table 2. Fixed Host Parameter

Fixed Host Parameter	**Default Values**
Number of LTM	10
Number of local transactions	Up to 500
Number of operation in each local transaction	6- 12
Probability of write operation for local transaction	0.3- 0.5
Lock time for each operation	10
Concurrency protocol	2 Phase Locking

Table 3. Mobile Host Parameters

Mobile Host Parameters	Default Values
Number of mobile units	500
Number of mobile sub transactions	1-10
Number of operations in each sub- transaction	6- 12
Probability of write operation	0.3 – 0.5
Probability of cash hit	0.2 – 0.8
Execution cost at the mobile host	2-5
Mobility value of mobile host number of visited MTM	5 – 50

7.3. Experiment Results and Discussion

The experiments are designed to study the impact of mobility ratio under different disconnection conditions on the performance measures to compare and contrast the different execution strategies. Figures 8 -13 show the impact of changing network conditions on the response time for both mobile and fixed transaction under different execution strategies. In each experiment, 1000 transactions are generated and 60% of these transactions are mobile. The mobility timer is varies from 5 to 50 seconds.

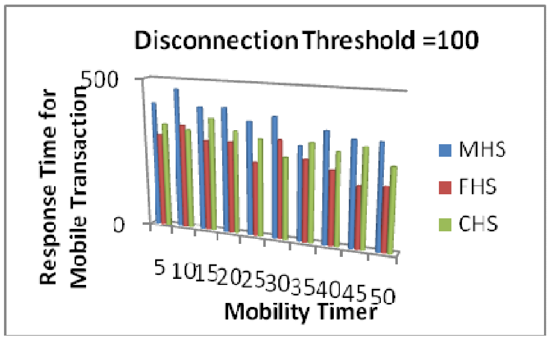


Figure 8: Response time for mobile transaction

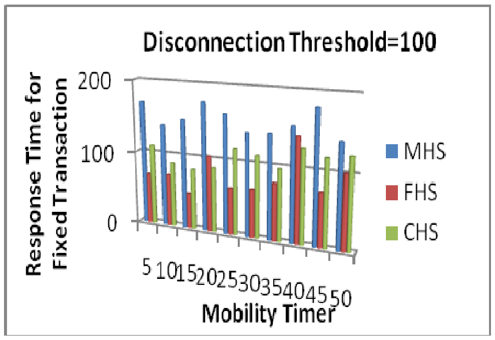


Figure 9: Response time for fixed transaction

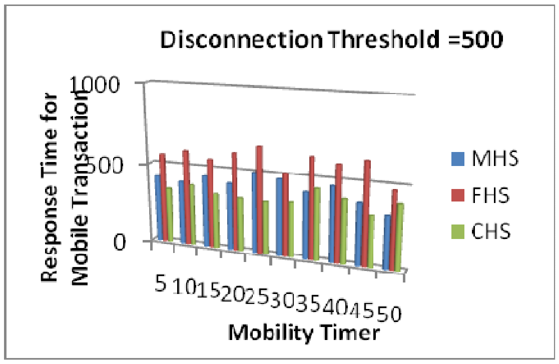


Figure 10: Response time for mobile transaction

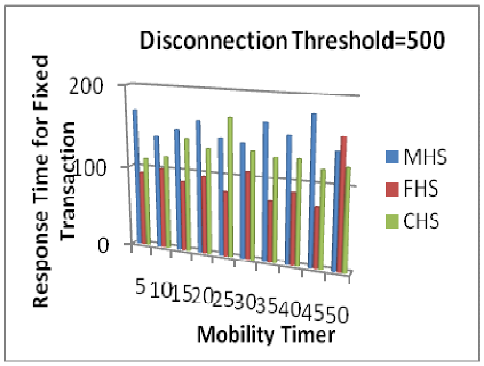
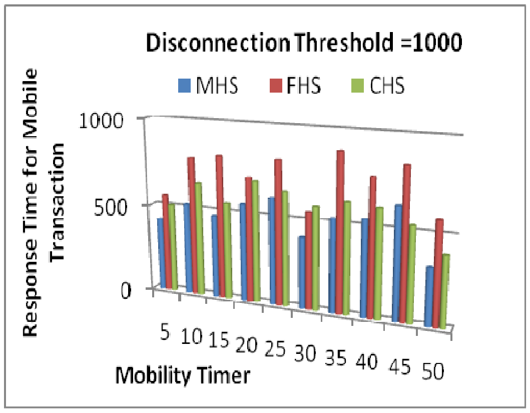


Figure 11: Response time for fixed transaction

Figure 8 shows the results when a mobile host has low disconnection (disconnection threshold is BW<=100). It can be seen that among the three approaches, the FHS outperform the other strategies, whereas the MHS is the worst. This is because a mobile subtransaction that schedules as a data request transaction needs more data transmission than if schedule as a transaction request. Moreover, a data request transaction takes a longer time for global commitment. This can be reflected in the response time for the fixed transaction as we can see in Figure 9. The CHS strategy is closer to FHS than to MHS which means that most of mobile subtransactions are scheduled as transaction requests. Even though the network is highly connected, the performance of the CHS may not outperform the FHS strategy. This is because the network connectivity is not the only factor which affecting the decision made by the CHS strategy to determine where the mobile subtransaction execution is take place, the size of data to be transmitted and the cash status also affecting the decision made by the CHS. So, if the cache status indicates that cached data for a transaction is available, the transaction may schedule as a data request transaction, even though the network is fully connected. Therefore, unlike the FHS strategy, the sub transactions of the mobile transaction may scheduled in a mixed matter based on the current state of the environment at that time. Figure10 and11 shows the results when a mobile host has medium disconnection (disconnection threshold is BW<=500) for both mobile and fixed transaction, respectively. It can be seen that the response time for all strategies are negatively affected by increasing the disconnection threshold. However, because of increasing disconnection probability, the performance of the FHS strategy becomes less than the other two strategies and CHS has the best performance. The reason is that when the network disconnection probability increases, the number of mobile subtarnsaction which schedule as data requests increase. On the other hand, the mobile subtransaction at connection time was scheduled as a transaction request, so this environment gives the CHS strategy more chance to exploit the information at the current environment state which makes it better than FHS at the disconnection time and also better than MHS at connection time. In Figure 11 the difference between the FHS and other strategies is more obvious than Figure 9. Since the disconnection probability is increases, The FHS strategy has more chance to block the basic transaction which comprises the first phase of any mobile subtransaction execution. This will decrease the blocking over head on the fixed transaction at the fixed host. On the other hand, in MHS and CHS strategies, a cached data can be used to execute the basic transaction. So, there is no need to block a mobile subtransaction if the cash status of the required data items is valid. As a consequence, the response time of mobile transactions decreases substantially when compared to the FHS approach. The advantages get by the mobile transaction under MHS and CHS in term of decreasing the response time will be negatively affect the fixed transaction as it have to wait for the basic transactions of the mobile subtransaction until finish its execution at the mobile host which is take a longer time than if the execution is take place at the fixed host. This can be seen in the Figure 12 and 13 which show the response time for both fixed and mobile transaction when there is a high disconnection, respectively.



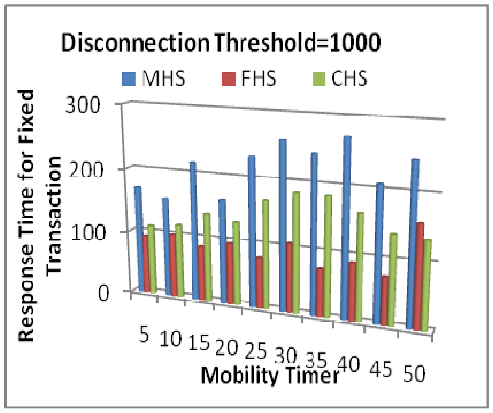


Figure 12: Response time for mobile transaction Figure 13: Response time for fixed transaction

The battery of mobile hosts considered to be one of a scarce resource in the mobile computing environment [14], an experiment is conducted to show the difference between three strategies in term of power consumption. Figures 14-16 show the simulation results for different mobility value with different disconnection thresholds. Since the whole mobile transaction take place at the fixed host under FHS strategy, only the communication overhead can affect the power consumption of the mobile host. This justifies the significant difference between FHS and other strategies Figure 14. On the other hand, the MHS has the worst effect on power consumption because all mobile transaction processing take place at the mobile host. The difference between the power consumption for the CHS and MHS strategies decrease as disconnection increases since there is more chance for mobile subtransaction to schedule as a data request transaction. Figure 14 shows that when the network is strongly connected.

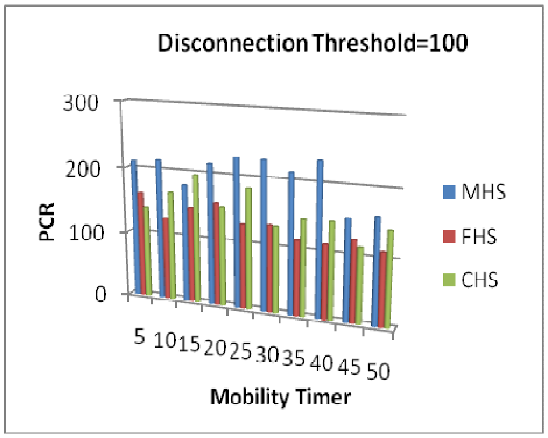


Figure 14: Power consumption

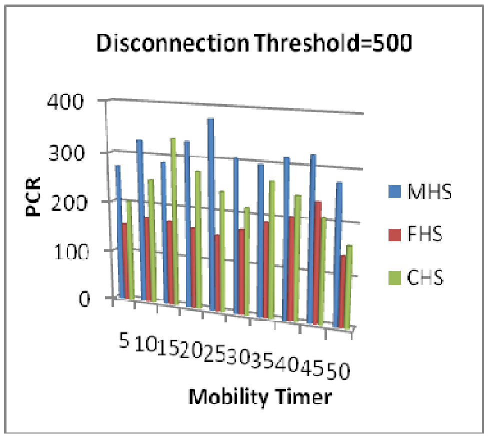


Figure 15: Power consumption

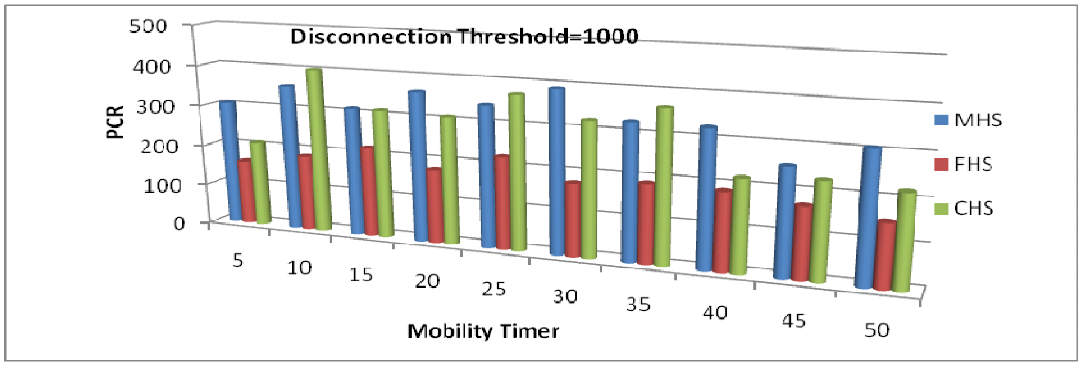
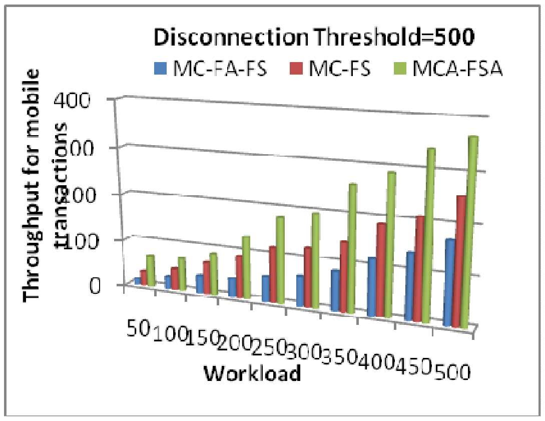


Figure 16: Power consumption

The MHS strategy takes more processing time on a mobile host than the CHS strategy. This is because, with the CHS, the number of mobile sub transactions which are scheduled as transaction requests increase so that the time spent on the mobile host is reduced. As the network disconnection increase the CHS strategy still has a better performance in reducing power consumption at the mobile host than MHS. However, as the network disconnection probability increase further, The Power consumption under CHS strategy approach the power consumption of the MHS strategy as we can see in Figure 16.



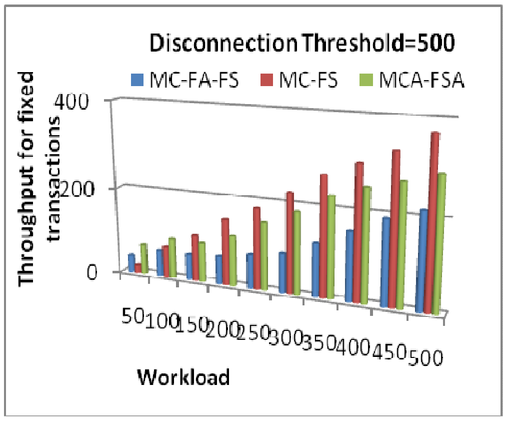
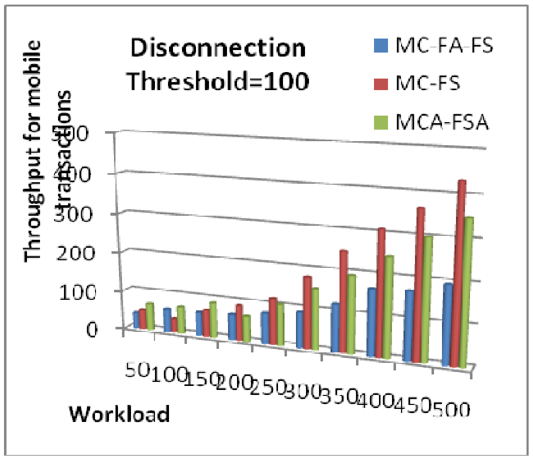


Figure 17: Throughput for mobile transactions

Figure 18: Throughput for fixed transactions

Figures 17-22 show the experimental results for system throughputs in terms of the number of completed transactions during a time interval using CHS strategies. The first experiment show the result under different workload assumptions where the mobility and the disconnected threshold are set to 20 and 500 respectively. The purpose of this experiment is to show how the MCA-FSA model gives better performance than other Computation models as workload increases at the average network connection for mobile transactions. As we can see in Figure 17 the MCA-FSA model consistently demonstrates better performance than MC-FA-FS and MC-FS computation models for a mobile transaction where Figure 18 show how the MCA-FSA model are close to the MC-FS computation model in term of fixed transaction throughput.



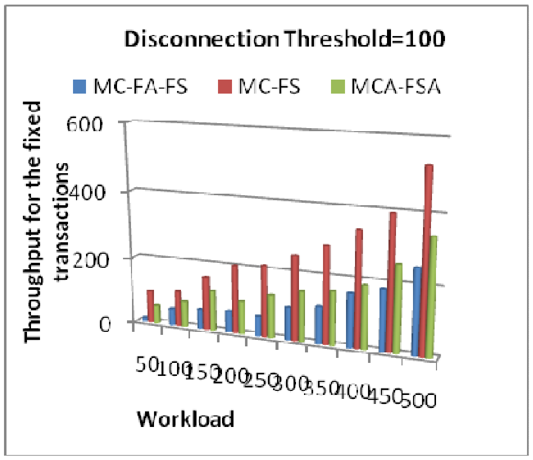
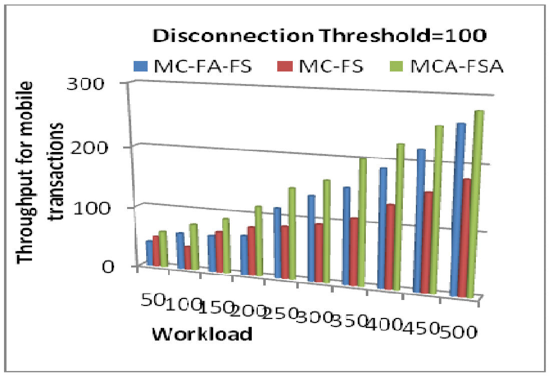


Figure 19: Throughput for mobile transaction

Figure 20: Throughput for fixed transactions

Figure 19 and 20 illustrates that with low disconnection (disconnection threshold =100); the MC-FS model has the highest throughput for both types of transactions where there is an improvement in fixed transaction throughput over the case of medium disconnection. While the MC-FA-FS the lowest and the MCA-FSA approach is in between the two. This result is consistent with the result shown in Figure 8 and 9 as an approach with a lower throughput can have a longer response time. However, Figure 21 shows degradation under all computation models with high disconnection probability for mobile transactions. The MCA-FSA can produce better throughput over the other two approaches because the basic transaction under the MC-FS model is unable to get through during network disconnection. At the same time, despite that the MC-FA-FS can carry on transaction processing with cached data it is ignoring the time interval during the connection period of the network. For the throughput of the fixed transaction, MC-FS model still have the superior over the other two computation models and the MCA-FSA become closer to the MC-FA-FS model than to MC-FS as we can see in the Figure 22.



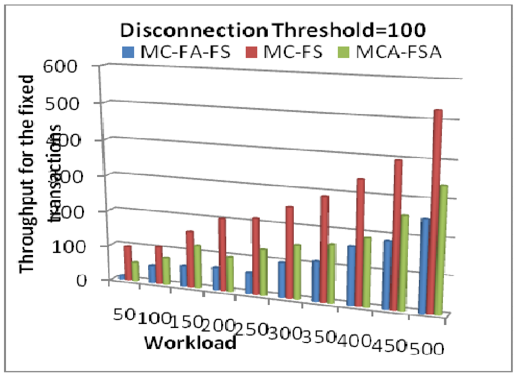


Figure 21: Throughput for mobile transactions Figure 22: Throughput for fixed transactions

Additional experiments were added Figure 23-25 that show the difference between different computation models over different execution strategies in this experiments, 50% of transactions were generated are mobile. The mobility timer is varies from 5 to 50 seconds and the workload is set to its default value with medium disconnection probability (disconnection threshold <=500); the result show that best gain of all is when we used the combined execution strategies CHS with MCA-FSA as the underling computation model.

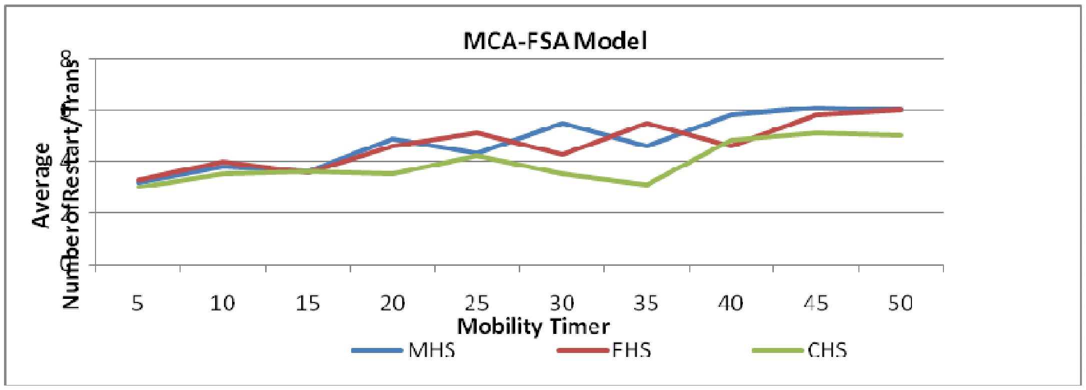


Figure 23: Different execution strategies under MCA-FSA model

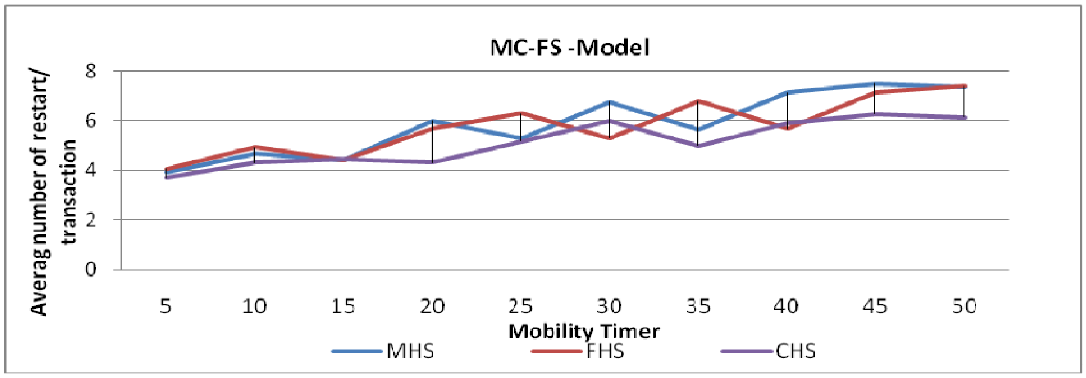


Figure 24: Different execution strategies under MC-FS model

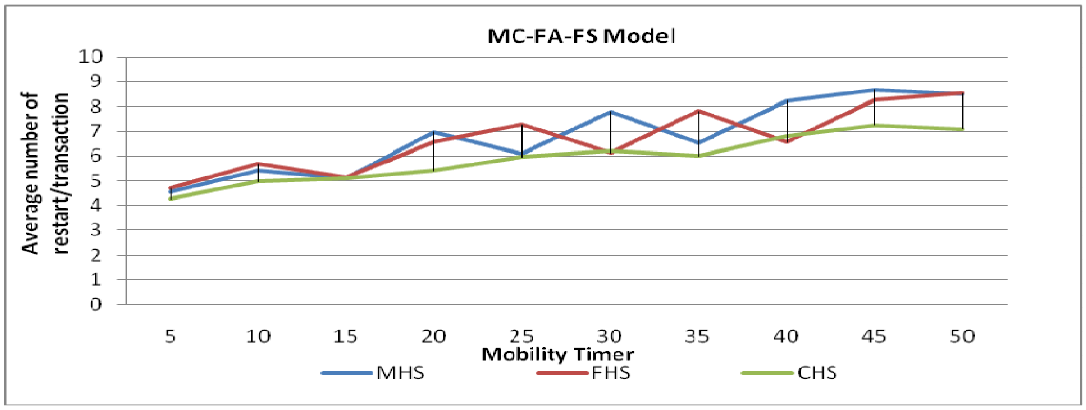


Figure 25: Different execution strategies under MC-FA-FS model

8. CONCLUSION

Mobility brings in new dimension to the existing solutions to the problems in distributed databases. We have reviewed some of the problems and existing solutions in that direction. We have highlighted the merits and demerits of existing solutions. We have found from the previous papers that, no comparative performance evaluation of models is presented. We observe that there is a need to investigate the properties of mobility, which can impact most the transaction processing. Also, there is a need to evaluate various transactions processing where a mixed fixed and mobile transactions are coexist in the system. In this paper we have introduce a frame work to investigate three transaction execution strategies based on the general assumption that the information of environment current state are available. To do so, we have built a simulator to imitate the mobile data base system with mixed transaction running on the system. To summarize, the simulation experiments performed in this paper investigate the performance of three transaction processing policies under the assumption that network disconnection occurs. The experiments are conducted from several perspectives by adjusting model parameters, such as, average inter-event time of bandwidth change and disconnection ratio. The simulation results show that if there is little or no network disconnection. The FHS has the shortest response time and the highest system throughput. This is not surprising because the system operates at or near the traditional fixed network environment. However, as network connectivity deteriorates, the MHS produces better system throughput and response time than the FHS. In general, the CHS approach produces better performance in terms of overall response time, elapsed processing time of a mobile host and total number of transactions completed by the system. Over all, the result show that best gain of all is when we used the combined execution strategies CHS with MCA-FSA as the underling computation model.

ACKNOWLEDGEMENTS

This research is funded by the Deanship of Research and Graduate Studies in Zarqa University /Jordan

REFERENCES

[1] I. Park and S.J. Hyun, A Dynamic Mobile Transaction Management Strategy for Data-intensive Applications based on the Behaviors of Mobile Hosts, Proceeding (367) Information Systems and Databases - 2002.

[2] P.K. Chrysanthis, Transaction processing in a mobile computing environment, in: Proceedings of IEEE workshop on Advances in Parallel and Distributed Systems, October, pp. 77-82, 1993.

[3] M.H. Eich, A. Helal, A mobile transaction model that captures both data and movement behavior, ACM/Baltzer Journal on Special Topics on Mobile Networks and Applications (1997).

[4] Q. Lu, M. Satyanaraynan, Improving data consistency in mobile computing using isolation only transactions, in: Proceedings of the Fifth Workshop on Hot Topics in Operating Systems, Orcas Island, Washington, May, 1995.

[5] S.K. Madria, B. Bhargava, System defined prewrites to increase concurrency in databases, in:Proceedings of First East- European Symposium on Advances in Databases and Information Systems (in co-operation with ACM-SIGMOD), St.-Petersburg, September, 1997.

[6] S.K. Madria, B. Bhargava, Improving availability in mobile computing using prewrite operations, Distributed and Parallel Database Journal (2001).

[7] Dirckze, L. Gruenwald, A toggle transaction management technique for mobile multidatabases, in: ACM Proceedings of International Conference on Information and Knowledge Management (CIKM), 1998.

[8] Pitoura, B. Bhargava, Building Information Systems for Mobile Environments, in: Proceedings of 3rd International Conference on Information and Knowledge Management, pp. 371-378, 1994,.

[9] K. Ramamritham, P.K. Chrysanthis, A taxonomy of correctness criterion in database pplications, Journal of Very Large Databases 4 (1) (1996).

[10] G.D. Walborn, P.K. Chrysanthis, Supporting semantics-based transaction processing in mobile database applications, in: Proceedings of 14th IEEE Symposium on Reliable Distributed Systems, September, 1995, pp. 31-40.

[11] E. Pitoura, B. Bhargava, Revising transaction concepts for mobile computing, in: Proceedings of the 1st IEEE Workshop on Mobile Computing Systems and Applications, December, pp. 164-168, 1994,

[12] A. Rasheed, A. Zaslavsky, Ensuring database availability in dynamically changing mobile computing environment, in: Proceedings of the 7th Australian Database Conference, Melbourne, Australia, 1996.

[13] E. Pitoura, B. Bhargava, Maintaining consistency of data in mobile computing environments, in: Proceedings of 15th International Conference on Distributed Computing Systems, June, 1995 (Extended version to appear in IEEE Transactions on Knowledge and Data Engineering, 1999).

[14] S.K. Madria, B. Bhargava, Improving availability in mobile computing using prewrite operations, Distributed and Parallel Database Journal (2001).

[15] P.K. Chrysanthis, Transaction processing in a mobile computing environment, in: Proceedings of IEEE workshop on Advances in Parallel and Distributed Systems, October, 1993, pp. 77-82.

[16] S.K. Madria, B. Bhargava, A transaction model for mobile computing, in: Proceedings 2nd IEEE International Database and Engineering Application Symposium (IDEAS'98), Cardiff, UK, 1998.

[17] S.K. Madria, B. Bhargava, on the correctness of a transaction model for mobile computing, in: 9th International Conference on Database and Expert System Applications (DEXA'98), Vienna, Austria, Lecture Notes in Computer Science, vol. 1460, Springer, Berlin, 1998.

[18] C. Pu, G. Kaiser, Hutchinson, Split-transactions for open-ended activities, in: Proceedings of the 14th VLDB Conference, 1988.

[19] L.H. Yeo, A. Zaslavsky, Submission of transactions from mobile workstations in a cooperative multidatabase processing environment, in: Proceedings of the 14th IEEE International Conference on Distributed Computing Systems (ICDCS'94), June, 1994.

[20] Ahmad alQerem, ” Impact of mobility on concurrent transactions mixture” Proceeding of the international conference on Computers, digital communications and computing Pages 116- 123 CDCC'11 USA ©2011 ISBN: 978-1-61804-030-5, 2011

[21] Z S. Madria, M. Baseer, V. Kumar, and S. Bhowmick, ",” Distributed An Efficient Concurrency Control Technique for Mobile Database Environment, Parallel Databases, vol. 22, pp. 165-196,2007.

[22] R. A. Dirckze and L. Gruenwald: A pre-serialization transaction management technique for mobile multidatabases, Mobile Networks and Applications (MONET), 5(4), 2000, pp 311-321.

[23] J. Holliday, D. Agrawal and A. E. Abbadi: Disconnection Modes for Mobile Databases, Wireless Networks, 8(4), 2002, pp 391-402.

[24] R. Hirsch, A. Coratella, M. Felder and E. Rodríguez: A Framework for Analyzing Mobile Transaction Models, Journal of Database Management, 12(3), 2001, pp 36-47.
[25] T. Kaneda, M. Shiraishi, T. Enokido and M. Takizawa: Mobile Agent Model for Transaction Processing on Distributed Objects, International Conference on Advanced Information Networking and Applications (AINA), 2004, pp506-511.
[26] atricia Serrano-Alvarado, Claudia Roncancio, Michel Adiba, "A Survey of Mobile Transactions", Distributed and Parallel databases, 16,193-230, 2004, Kluwer Academic Publishers.
[27] V. Kumar, N. Prabhu, M. H. Dunham and A. Y. Seydim: TCOT-A Timeout- Based Mobile Transaction Commitment Protocol, IEEE Transactions on Computers, 51(10), 2002, pp 1212-1218.
[28] D. L. R. Salman Abdul Moiz, "A Real Time Optimistic Strategy to achieve Concurrency control in Mobile Environments using ondemand multicasting," International Journal of Wireless & Mobile Networks (IJWMN), vol. 2, pp. 172-185, 2010.
[29] S. Madria, M. Baseer, V. Kumar, and S. Bhowmick, 2007"A transaction model and multiversion concurrency control for mobile database systems," Parallel Databases, vol. 22, pp. 165-196,.
[30] Ahmad al-Qerem, "Framework for Transaction Execution Strategies in Mobile Data Base Systems," International Journal of Electronics and Electrical Engineering, Vol. 1, No. 1, pp. 15-18, March 2013. doi: 10.12720/ijeee.1.1.15-18.
[31] Venkatraman, S, "Mobile Computing Models - Are they Meeting the Mobile Computing Challenges?" Association for Computing Machinery New Zealand Bulletin, 1 (1) (ISSN 1176-9998), 2005.

AUTHOR

Ahmad Alqerem obtaining a BSc in 1997 from JUST University and a Masters in computer science from Jordan University in 2002. PhD in mobile computing at Loughborough University, UK in 2008. He is interested in concurrency control for mobile computing environments, particularly transaction processing. He has published several papers in various areas of computer science. After that he was appointed a head of internet technology Depts. Zarka University