Global Transaction Models on Mobile Computing

[1]V Shiva Narayana Reddy, [2]A Venkat Reddy, [3]P Mahesh Kumar, [4]B Seshagiri
[123]VBITS, Nadergul, Hyderabad, AP, India
[4]JIET, IBP, Hyderabad, AP, India

Abstract
The issues mobile computing paradigm introduces new technical properties in the area of database systems. However, techniques for traditional distributed database management have been based on the assumption that the location of end connections among hosts in the distributed system does not change. Distributed systems are expected to support mobile computations executed over a network of fixed and mobile hosts. On the other hand, in mobile computing, these assumptions are no longer valid and mobility of hosts creates a new kind of locality that migrates as hosts move. Consequently, existing solutions for traditional distributed database management may not be applicable directly to the mobile computing environment. Many research proposals that focus on supporting transaction processing models in mobile computing environments have been developed. However, there are still major issues that have not been completely solved. One of the problems is to support the requirement for mobile transaction processing system. Here in this paper, I have presented a comparative analysis of existing mobile transaction models on the basis of reporting transactions.

Keywords
Transaction Properties, Mobile Transaction, Virtual Memory

I. Introduction

A transaction is nothing but a legitimate implementation of database operation [1]. Users can interact with the database by one or many database operations. The database operations can be gathered together to form a unit of execution program that is called a transaction. A transaction starts from creating a coherent state of database [2]. A transaction transforms the database from one consistent state to another consistent state. A distributed transaction processing system is a collection of sites or nodes that are connected by communication networks. A mobile host can be disconnected from the database servers for long periods; therefore, transactions that are executed at the mobile host may suffer from long blocking if the necessary data is not available at the mobile host. To deal with this problem, the mobile transaction processing system should have the capacity to maintain virtually so that it can carry out sufficient data for performing the transactions while being disconnected from the database servers.

II. Prior Literature

Lars Frank [1999], in this paper the author has focused on implementing the global semantic ACID property in systems using mobile computing. The global atomicity property has been implemented by using retainable, pivot and compensable sub-transactions in that order. The global consistency property has been managed by the application programs themselves supported by tools. The global isolation property has been implemented by using counter measures to the missing isolation of the updating transactions and the global durability property has been implemented by using the durability property of the local DBMS systems. To implement ACID properties, a model called Countermeasure Transaction Model has been introduced in this paper. In The Countermeasure Transaction Model a global transaction consists of a root transaction (client transaction) and several single site sub-transactions (server transactions). The sub-transactions themselves can be nested transactions; i.e. a sub-transaction may be a parent transaction for other sub-transactions. Tools used to access remote sub-transactions are: Remote Procedure Call, Update Propagation, and Transaction Message. This Model ensures ACID properties in multi-database system [3].

Hossam S. Hassanein [2000], in this paper, the focus is to study the effects of transactions characteristics on system performance. A detailed simulation model is developed and conducted several experiments to measure the impact of transactions characteristics on the performance. First, the effect of the number of leaves on the performance of nested transactions is investigated under different shaping parameters. Also, effects of the depth of the transaction tree on the system performance are investigated. This paper introduced a comprehensive simulation model for studying the performance of nested transactions in database systems. The model was used to investigate the performance effects of two main factors on a system with nested transactions: the number of levels and the number of leaves of the transaction tree. It was shown that, in general, increasing the number of leaves improves the performance of the system. This was observed for any number of levels, transaction size, and data contention level and leaf arrival time, with or without common items. However, at high data contention levels and large transaction sizes, increasing the number of leaves beyond a certain limit may cause performance degradation. This is due to the significant increase in the restart ratio caused by increasing the number of leaves. It is also shown that increasing the number of levels of nested transactions degrades the system performance for nested transactions of small number of leaves. The effect of the number of levels on the performance of nested transactions of large number of leaves is insignificant, especially at high multi-programming levels [4].

Lisa Clark, Omer Erdem Demir [2003], this paper, presents a survey of transaction management models for wireless and mobile databases. Comparison of the models has been presented in this paper and comparison shows that the execution models are designed for specific network and data distribution topologies. They also relax the ACID to make the models more responsive and to avoid the deadlocks. Some of the models also define transactions of different consistency levels [5].

III. Mobile Transaction Models

A. Kangaroo transaction model (KTM)

1. Description

The Kangaroo transaction model [5-6] is designed to capture the movement behavior and the data behavior of transactions when a mobile host moves from one mobile cell to another. This transaction model is built based on the concepts of global and split transactions in a heterogeneous and multi-database environment. The global transaction is split when the mobile host moves from one mobile cell to another, and the split transactions are not joined back to the global transaction. The Kangaroo transaction model assumes that the mobile transactions may start and end at different

 ISSN : 0976-8491 (Online) | ISSN : 2229-4333 (Print)

locations. The characteristics of the Kangaroo transaction model are:

- Mobile transactions that include a set of sub-transactions called global and local transactions are initiated by mobile hosts. These mobile transactions are entirely executed at the local database servers that reside on the fixed and wired connected networks.
- The execution of a Kangaroo sub-transaction in each mobile cell is supported by a Joey transaction that operates in the scope of the mobile support station. The Joey transaction plays role of a proxy transaction to support the execution of the sub transactions of the Kangaroo transaction in the mobile cell.
- The movement of the mobile host from one mobile cell to another is captured by the splitting of the on-going Joey transaction at the old mobile support station and the creating of new Joey transaction at the new mobile support station. The execution of the Joey transaction is supported by the Data Access Agents (DAA) that act as the mobile transaction managers at the mobile support stations.

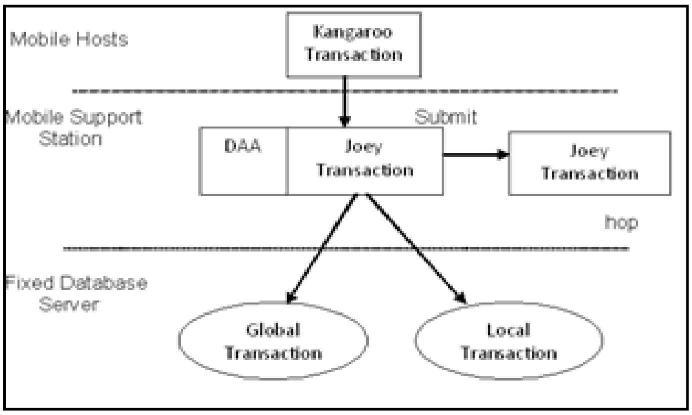

Fig. 1: Kangaroo Transaction Model

2. Transaction Properties

The Kangaroo transaction is the basic unit of computation in mobile environments. The serializability of mobile transactions is not guaranteed, and there is no dependency among Joey transactions, i.e., each Joey transaction can commit independently. Two transaction processing modes, which are compensating and split modes, are supported by the model. For compensating mode, when a failure occurs, the entire Kangaroo transaction is undone by executing compensating transactions for all those Joey transactions. For split mode, the local DBMS takes responsibility for aborting or committing sub-transactions.

3. Mobility

The Kangaroo transaction model keeps track of the movement of mobile hosts via the support of the DAA that operates at the mobile support station. In other words, the mobility of mobile hosts is captured on the condition that the mobile hosts always may communicate with the mobile support stations. While mobile hosts move from one mobile cell to another, the hand-off processes are carried out by the DAAs.

4. Disconnection

Disconnected transaction processing is not considered in Kangaroo transaction model. The processing of Kangaroo transactions is entirely moved to the fixed database servers for executing.

5. Distributed Execution

The mobile transactions are initiated at the mobile hosts, and entirely executed at fixed hosts. Transaction results are forwarded back to the mobile hosts. The Kangaroo transaction model has shown that the structure of mobile transactions at the specification and execution phases (with the dynamic support of Joey transactions) can be different because of the mobility behavior, i.e., fast or slow movements, of the mobile host.

B. Reporting and Co-Transaction Model (RCTM)

1. Description

Reporting and Co-transactions transaction model [7] is based on a two level nested transaction model. A reporting transaction TR shares its partial results to top-level transaction S by delegating its operations. The delegation process can happen at any time during the execution of transaction TR. A co-transaction is a reporting transaction but it cannot continue executing during the delegation process. Thus, the co-transaction behaves as a co-routine, and resumes execution when the delegation process is completed.
This model arranges the mobile transaction into following four types:

(i). Atomic Transactions

It is related with substantial events the normal aborts and commits properties.

(ii). Non-Compostable Transactions

It is not linked with compensating transaction. It can execute at any time and the parents of these transactions have the responsibility to commit and abort [8].

(iii). Reporting Transactions

A report can be regarded as a delegation of state between transactions. The reporting transaction not assigning all its results to its parent transactions .It only has one receiver at any time during execution. The updating is completed permanently if receiving parent transaction is successfully executed but if receiver parent transactions unsuccessfully terminate then corresponding reporting transaction abort.

(iv). Co-Transactions

These transactions executed like co-procedures executed. When one transaction is executed then control passes from current transaction to another transaction during sharing the results. At a time either both transaction successfully executed or failed.

2. Transaction Properties

The top-level transaction is the unit of control, and atomic sub transactions are compensable transactions. A Reporting transaction that is compensable does not have to delegate all of the committed results to the top-level transaction when it commits. Sub-transactions that are non-compensable delegate all of their operations to the top-level transaction when it commits.

3. Mobility

The locations of mobile hosts are determined via the identification of mobile support stations. However, the model does not mention explicitly what happens when mobile hosts move from one mobile cell to another.

4. Disconnection

Delegation operations require a tight connectivity between the delegator (i.e., Report and Co-transaction) transactions and the delegate transaction (i.e., the top level transaction). Therefore, disconnection is not supported in this model.

5. Distributed Execution

The model supports distributed transaction processing among mobile hosts and fixed hosts where the network connectivity among these hosts is assumed to be available when it is needed.

C. Pro-Motion Transaction Model (PMTM)

1. Description

The Pro-motion transaction model [6] is a nested transaction model. The Pro-motion model focuses on supporting disconnected transaction processing based on the client-server architecture [9]. Mobile transactions are considered as long and nested transactions where the top-level transaction is executed at fixed hosts, and sub transactions are executed at mobile hosts. The execution of sub-transactions at mobile hosts is supported by the concept of compact objects.

Methods common to all compacts		Type specific Methods
Obligations	Data	Consistency Rules
State Information		

Fig. 2: Compacts as Objects

Compact objects are constructed by compact manager at database servers. Necessary information is encapsulated within a compact object. The compact objects are co-managed by the compact managers (resided at the database servers), the mobility managers (at base host), and the compact agents (at the mobile hosts). The compact object plays a role as a contractor that supports data replication and consistency between mobile hosts and database servers. When a mobile host is disconnected, the compact agent takes responsibility for managing all local database operations of mobile transactions at the mobile host. When the mobile host reconnects to database servers, the compact objects are verified against global consistency rules before the locally committed mobile transactions are allowed to commit. Fig. 3 shows the architecture of the Pro-motion transaction model. Transaction processing consists of four phases: hoarding, disconnected, connected, and resynchronization [10]. Shared data is downloaded to the mobile host in the hoarding phase. When the mobile host is disconnected from the fixed host, transactions are disconnectedly executed at the mobile host. If the mobile host connects to the fixed database, the transactions are carried out with the support of the compact manager. When the mobile host reconnects to a fixed host, the results of local transactions are synchronized with the database.

2. Transaction Properties

The Pro-motion transaction model supports ten different levels of isolation. Transactions are allowed to locally commit at mobile hosts; the committee results of these transactions are made available to other local transactions. However, the local committed results must be validated when the mobile hosts reconnect to the database servers. Therefore, the durability property of transaction is only ensured when the transaction results are finally reconciled at the fixed database.

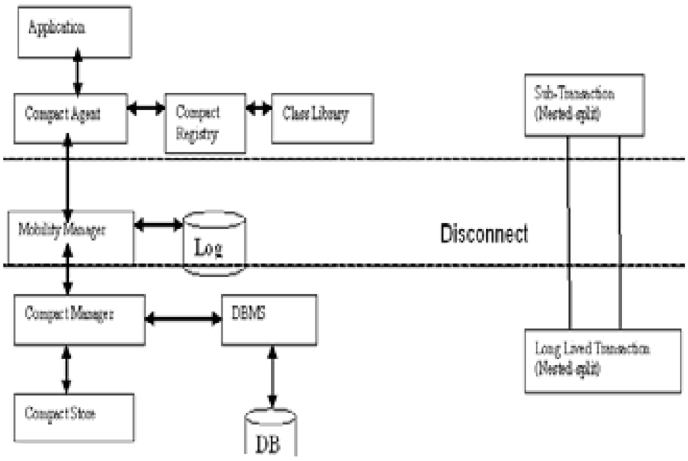

Fig. 3: Pro-Motion Transaction Architecture

3. Mobility

Though the mobility manager supports communications between the mobile host and the database servers, however in the Pro-motion transaction model the feature of mobility is not explicitly discussed.

4. Disconnection

Pro-motion transaction model supports disconnected transaction processing via the support of compact objects. When the mobile host is disconnected from the fixed database, the sub-transactions are split and executed at the mobile host (these split sub-transactions are not joined when the mobile host reconnects to the fixed database). Disconnected transaction processing is a dominant transaction processing mode in Pro-motion even when the mobile hosts are able to connect to the database server. Therefore, the Pro-motion transaction model requires high-capacity mobile resources at the mobile hosts.

5. Distributed Execution

Transactions are mostly executed at mobile hosts and the results are reconciled at the database servers. Therefore, the distributed transaction processing is not strongly supported by the model.

D. Two - Tier Transaction Model (2TTM)

1. Description

The two-tier (also called Base-Tentative) transaction model is based on a data replication scheme. For each data object, there is a master copy and several replicated copies. There are two types of transaction: Base and Tentative. Base transactions operate on the master copy; while tentative transactions access the replicated copy version. A mobile host can cache either the master or the copy versions of data objects. While the mobile host is disconnected, tentative transactions update replicated versions. When the mobile host reconnects to the database servers, tentative transactions are converted to base transactions that are re-executed on the master copy. If a base transaction does not fulfill an acceptable correctness criterion (which is specified by the application), the associated tentative transaction is aborted.

 ISSN : 0976-8491 (Online) | ISSN : 2229-4333 (Print)

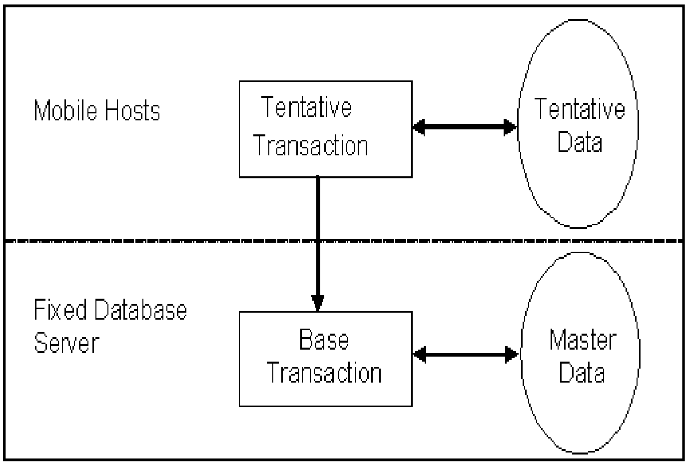

Fig. 4: Two-Tier Transaction Model

2. Transaction Properties

Tentative transactions locally commit at the mobile host on replicated copies, and the committed results are made visible to other tentative transactions at that mobile host. The final commitments of those tentative transactions are performed at the database servers.

3. Mobility

Two-tier transaction model does not support the mobility of transactions.

4. Disconnection

While the mobile hosts are disconnected from the database servers, tentative transactions are locally carried out based on replicated versions of data objects.

5. Distributed Execution

Two distinct transaction execution modes are supported: connected and disconnected. Transactions are tentatively carried out at disconnected mobile hosts, and re-executed as base transactions at the database servers.

E. Weak-Strict Transaction Model (WSTM)

1. Description

The Weak-Strict (also called Clustering) transaction model consists of two types of transaction: weak (or loose) and strict [9]. These transactions are carried out within the clusters that are the collection of connected hosts which are connected via high-speed and reliable networks [11]. In each cluster, data that is semantically related is locally replicated. There are two types of a replicated copy: local consistency (weak) and global consistency (strict). The weak copy is used when mobile hosts are disconnected or connected via a slow and unreliable network. Weak and Strict transactions access weak and strict data copies, respectively. Fig. 5 presents the architecture of this transaction model. When mobile hosts reconnect to database servers, a synchronization process reconciles the changes of the local data version with the global data version.

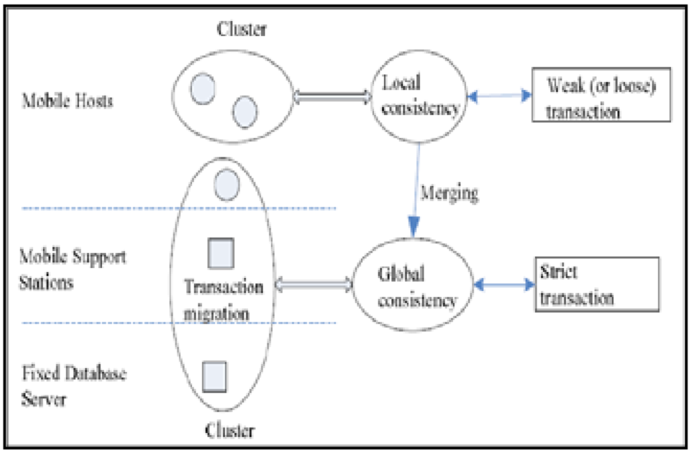

Fig. 5: Architecture of Weak-Strict Transaction Model

2. Transaction Properties

Weak transactions are allowed to commit within its cluster, and results are made available to other local weak transactions. When mobile hosts are reconnected, the results of weak transactions are reconciled with the results of strict transactions. If the results of a weak transaction do not conflict with the updates of strict transactions, weak transactions are globally committed; otherwise they are aborted.

3. Mobility

The concept of transaction migration is proposed to support the mobility of transactions, and to reduce the communication cost. When the mobile host moves and connects to a new mobile support station, parts of the transaction that are executed at the old mobile support stations are moved to the new one. However, no further details about the design or implementation are given.

4. Disconnection

The Weak-Strict transaction model supports transaction processing in disconnected and weakly connected modes via weak transactions.

5. Distributed Execution

Transaction execution processes can be distributed between the mobile host and the database servers within a cluster that the mobile host participates in. However, the distributed transaction processing among mobile hosts in a cluster is not discussed.

F. Pre-Serialization Transaction Model (PSTM)

1. Description

Pre-serialization transaction model [5] is built on top of local database systems. Mobile transactions (also called global transactions) are submitted from mobile hosts through the global transaction coordinators that reside at the mobile support stations. This mobile transaction is entirely processed at local database systems (As shown in Fig. 6). At each node (or site), there is a site manager that administrates all the transactions executed at that node. When a global transaction is prepared to commit, a global transaction coordinator will carry out an algorithm, called Partial Global Serialization Graph algorithm that detects any non-serializable schedule among the mobile transactions. If there is a cycle in the graph, i.e., the schedule is non-serializable, the mobile transaction is aborted.

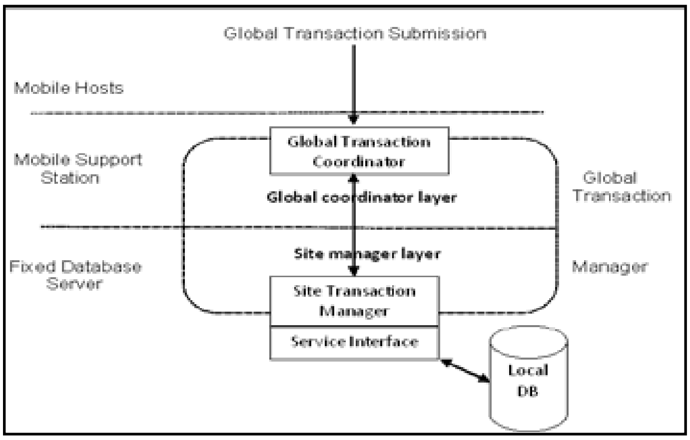

Fig. 6: Pre-serializable Transaction Model

2. Transaction Properties

Each sub-transaction of a global transaction is managed by the local transaction manager. The global serializable graph of transactions is constructed by collecting sub-graphs from the local sites. The atomicity property of the global transaction is relaxed by the concepts of vital and non-vital sub-transactions.

If a vital sub-transaction aborts, its parent transaction must abort. However, the parent transaction does not abort if a non-vital sub-transaction aborts. When a sub-transaction commits at the local database system, the results are made visible to other transactions at this local database system.

3. Mobility

The global transaction coordinators that reside at the mobile support stations support the mobility of mobile transactions. This is done by transferring the global data structure from one global transaction coordinator to another as the mobile host moves from one mobile cell to another.

4. Disconnection

Mobile transactions are submitted from a mobile host, and sub transactions are executed at local database servers. When the mobile host is disconnected, the global transaction is marked as disconnected if the disconnection is known and planned. The execution of the global transaction is still carried out at the local database servers. On the other hand, if the disconnection is unplanned, the global transaction is suspended. The global transaction is resumed when the mobile host reconnects to the mobile support station.

5. Distributed Execution

Mobile transactions are submitted from mobile hosts, and the entire transactions are distributed among local database servers through the support of mobile support stations. The mobile hosts do not take part in the execution processes.

G. Moflex Transaction Model (MTM)

1. Description

The Moflex transaction model [12-13] is an extension of the Flex transaction model to support mobile transactions. The Moflex model is built on top of multi-database systems and based on the concepts of split-join transactions. The main characteristics of a Moflex transaction are:

- A Moflex transaction that consists of compensable or non-compensable sub transactions is initiated by the mobile host. These sub-transactions are submitted to the Mobile Transaction Manager (MTM) that resides at the mobile support station. The MTM will send these sub-transactions to the local execution monitor (LEM) at local database systems for executing [14]. Fig. 7 presents the architecture of Moflex transaction model.
- Each Moflex transaction T is accompanied by a set of success and failure transaction dependency rules, hand-over control rules, and acceptable goal states. Dependent factors that include the execution time, cost and execution location of transactions are also specified in the definition of the Moflex transaction. Furthermore, joining rules are provided to support the join of the split sub-transactions (sub-transactions are split when the mobile host moves from one mobile cell to another).

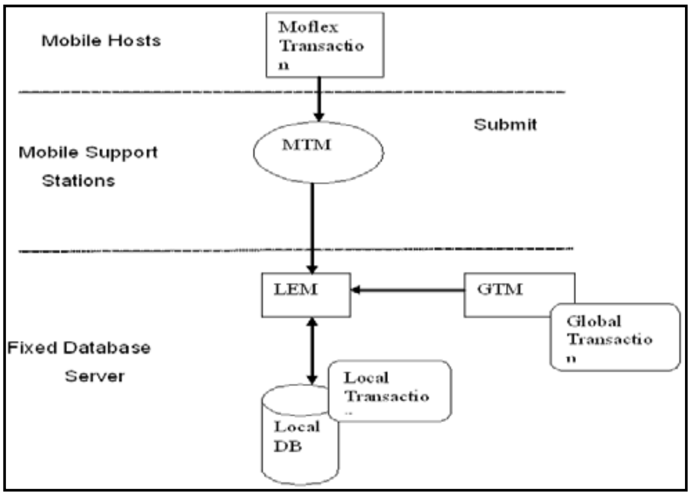

Fig. 7: Architecture of Moflex Transaction Model

2. Transaction Properties

The mobile transaction managers make use of the two-phase commit protocol to coordinate the commitment of the Moflex transaction. The Moflex transaction commits when its sub-transactions that are managed by MTM have reached one of the acceptable goal states, otherwise it is aborted. A compensable sub-transaction is locally committed, and the results are made visible to other transactions. For non compensable sub-transactions, the last mobile transaction manger, which corresponds to the end location of the mobile host, plays the role as the committing coordinator.

3. Mobility

The mobility of transactions is handled by splitting the sub-transaction, which is executed on the local database at the current mobile cell, as the mobile host moves from one mobile support station to another (with the support of the mobile transaction manager). Hand-over control rules must be specified for each sub-transaction. If a sub-transaction is compensable and location independent, it will be split into two transactions; one will continue and commit at the current local database, the second will be resumed at the new location. If the sub-transaction is location dependent, at the new location, the sub-transaction must be restarted. If a sub-transaction is non compensable, the sub-transaction is either restarted as a new one in the mobile cell if it is location dependent, or continued if it does not depend on the location of the mobile host.

ISSN : 0976-8491 (Online) | ISSN : 2229-4333 (Print)

4. Disconnection
Moflex transaction model does not support disconnected transaction processing. The Moflex transaction model requires network connectivity between the mobile host and the mobile support stations during the execution process.

5. Distributed Execution
The execution of a Moflex transaction is transferred to local database systems at fixed hosts to be carried out there. Moflex transaction model provides a framework to specify the execution of transactions in mobile environments. The main drawback of the Molex transaction model is that the specification of mobile transactions must be fully specified in advance, therefore, the Moflex transaction model may not have the capacity to deal with un-expected or un-planned situations.

Table 1: Comparative Study of Some Selected Existing Mobile Transaction Models

Model Name	Mobility	Disconnection	Distributed Execution	Heterogeneity
Kangaroo transaction model	Yes Partially	No	Yes	No
Reporting and Co-transaction model	No	No	Yes	No
Pro-motion transaction model	No	Yes	No	No
Two-Tier(Base - Tentative) transaction model	No	Yes	Yes	No
Weak-Strict (Clustering)	Yes Partially	Yes	Yes	No
Pre-serialization transaction model	Yes	Planned-Yes Unplanned-No	No	No
Moflex transaction model	Yes	No	No	No

IV. Conclusion
In this paper, the conclusion has been stated that some of the selected existing mobile transaction models support multiple issues like mobility, disconnection, distributed execution, transaction properties and the transaction over here is done Virtually, so the researchers can do work on this issue by incorporating new models or by proposing a existing model.

References

[1] P. K. Chrysanthis,"Transaction Processing in Mobile Computing Environment", IEEE Workshop on Advances in Parallel and Distributed Systems, pp. 77-83, 1993.

[2] D. Barbara,"Mobile Computing and Databases–A Survey", IEEE Transactions on Knowledge and Data Engineering (TKDE), 11(1), pp. 108-117, 1999.

[3] P. K. Reddy, M. Kitsuregawa,"Speculative Lock Management to Increase Concurrency in Mobile Environments", International Conference on Mobile Data Access (MDA), 1999, pp. 82-96.

[4] K. A. Momin, K. Vidyasankar,"Flexible Integration of Optimistic and Pessimistic Concurrency Control in Mobile Environments Advances in Databases and Information Systems - Database Systems for Advanced Applications (ADBIS-DASFAA)", 2000, pp. 346-353.

[5] Lisa Clark, Omer Erdem Demir,"Transaction Management in Mobile Distributed Databases", 2003.

[6] Lars Frank,"Atomicity Implementation in Mobile Computing", in Proceeding of 10th International Workshop on Database and Expert System Application , pp. 105, Sep 1999.

[7] G. Alonso, H. F. Korth,"Database System Issues in Nomadic Computing", in Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 388 – 392, 1993.

[8] M. Satyanarayanan,"Fundamental Challenges in Mobile Computing", in Proceedings of fifteenth annual ACM symposium on Principles of Distributed Computing, pp. 1-7, 1996.

[9] Ayse Yasemin Seydim,"An Overview of Transaction Models in Mobile Environments", 1997.

[10] Jim Gray, Andreas Reuter,"Transaction Processing: Concepts and Techniques", Morgan Kaufmann Publishers, 1993.

[11] W. Booth, G. G. Colomb, J. M. Williams,"Transaction Processing Models in Wireless Network", The Craft of Research, University of Chicago Press, 1995.

[12] Sharmila John Francis, Elijah Blessing Rajsingh, "Performance Analysis of Clustering Protocols in Mobile Ad hoc Networks", Journal of Computer Science 4(3), pp. 192-204, 2003.

[13] E. Pitoura, G. Samaras,"Data Management for Mobile Computing", Kluwer Academic Publishers, 1998.

[14] T. Tmiclinski, B. R. Badrinath,"Data management for mobile computing", Sigaiod Rfcord, 22(1), pp. 34-39, March 1993.

V.Shiva Narayana Reddy received his B.Tech degree in Computer Science & Information Technology, M.Tech degree in Computer Science from JNTU, AP, India, He was an Assistant Professor, Associate Professor, with Department of Computer Science and Engineering under JNTU Affiliated colleges from the year of 2005. Presently he is working as an Associate professor, at VBIT&S from 2011 which is located at Hyderabad. His research interests include Data Compression on Group Movement patterns, Ethical Hacking. At present, he is working on the emerging issues of software engineering.