ISSN: 2229– 6948 (ONLINE)
DOI: 10.21917/ijct.2010.0030

AN AGENT BASED TRANSACTION PROCESSING SCHEME FOR DISCONNECTED MOBILE NODES

J.L. Walter Jeyakumar[1] and R.S. Rajesh[2]
Department of Computer Science and Engineering, Manonmaniam Sundaranar University, Tirunelveli, India
E-mail: [1]walterjeya@hotmail.com, [2]rs_rajesh1@yahoo.co.in

Abstract
We present a mobile transaction framework in which mobile users can share data which is stored in the cache of a mobile agent. This mobile agent is a special mobile node which coordinates the sharing process. The proposed framework allows mobile affiliation work groups to be formed dynamically with a mobile agent and mobile hosts. Using short range wireless communication technology, mobile users can simultaneously access the data from the cache of the mobile agent. The data Access Manager module at the mobile agent enforces concurrency control using cache invalidation technique. This model supports disconnected mobile computing allowing mobile agent to move along with the Mobile Hosts. The proposed Transaction frame work has been simulated in Java 2 and performance of this scheme is compared with existing frame works.

Keywords:
Transaction, Concurrency Control, Mobile Database, Cache Invalidation, Mobility

1. INTRODUCTION

The recent advancements in portable devices, wireless network technology and satellite services have led to the development of a unique mobile computing environment. This environment will allow the mobile users to retrieve the desired information from anywhere at any point of time. With the evolution of PCS (Personal Communication System) and GSM (Global System for Mobile communication), advanced wireless communication services are being offered to the mobile users. The proposed Mobile Database System is a distributed client/server system based on PCS or GSM in which clients can move around freely while performing their data processing activities in connected or disconnected mode. Frequent disconnections, mobility, limited battery power and resources pose new challenges to mobile computing environment. Frequent aborts due to disconnection should be minimized in mobile transactions. Correctness of transactions executed on both fixed and mobile hosts must be ensured by the operations on shared data. Blocking of mobile transactions due to long disconnection periods should be minimized in order to reduce communication cost and to increase concurrency. After disconnection, the mobile host should be able to process transactions and commit locally. In Mobile computing, there is always a competition for shared data since it provides users with the ability to access information through wireless connections that can be retained even while the user is moving. Further, mobile users are required to share their data with others. This provides the possibility of concurrent access of data by mobile hosts which may result in data inconsistency. Concurrency control methods have been used to control concurrency. Due to limitations and restrictions of wireless communication channels, it is difficult to ensure consistency of data while sharing takes place.

In this paper, we propose a mobile transaction frame work that will allow mobile users to share data cached in the Mobile Agent which is a special node for coordinating the sharing process. Whenever an MH enters into a Mobile Agent area it can connect and access the data in the cache. But upon update request by a MH, updation is done at the local cache and invalidation report is sent to all the mobile hosts which have already accessed the same data. This will force the mobile hosts to refresh their data values. This framework also provides a provision for transaction update during disconnect. Data Access Manager at the Mobile Agent will take care of concurrency control while sharing takes place. Concurrency control is enforced using cache invalidation technique. We also take into account the mobility of the Mobile Hosts along with the Mobile Agent.

The remaining part of this paper is organized as follows. Section 2 summarizes the related research. Section 3 focuses on the Mobile Agent based architecture. Section 4 specifies the proposed framework for disconnected mobile computing. Section 5 gives the performance analysis and in Section 6 the conclusion is presented.

2. RELATED WORK

When simultaneous access to data is made at the server, concurrency control techniques are employed to avoid data inconsistency. Conventional locking based concurrency control methods like centralized Two Phase locking and distributed Two Phase locking are not suitable for mobile environment. In centralized Two Phase locking scheme [1], where one node is responsible for managing all locking activities, the problem of single point failure cannot be avoided. The distributed two phase locking scheme used in [2], allows all nodes to serve as lock managers. But in the event of data partition, this algorithm could degenerate into a centralized two phase scheme. In conventional locking scheme, the communication overhead that arises due to locking and unlocking requests can create a serious performance problem because of low capacity and limited resources in mobile environment [3]. Moreover, it makes mobile hosts to communicate with the server continuously to obtain and manage locks [4].

The timestamp approach for serializing the execution of concurrent transactions was developed for the purpose of more flexibility, to eliminate the cost of locking, and to cater to the requirements of distributed database systems [5], [6]. In timestamping, the execution order of concurrent transactions is defined before they begin their execution. The execution order is established by associating a unique timestamp to every

transaction. When two transactions conflict over a data item, their timestamps are used to enforce serialization by rolling back one of the conflicting transactions. To exploit the dynamic aspects of two phase locking and the static ordering of timestamping, a number of concurrency control techniques were developed using a combined approach. In mixed approach techniques called Wound-wait and Wait-die [5], [7], the enforcement of mutual exclusion among transactions is carried out using locking while conflicts are resolved using timestamps.

In [8], optimistic concurrency control scheme is used to minimize locking overhead by delaying lock operation until conflicting transactions are ready to commit. They rely on efficiency in the hope that conflicts between transactions will not occur. Without using lock during the execution of the transactions, this scheme promotes deadlock free execution. In optimistic concurrency control with dynamic time stamp adjustment protocol, client side write operations are required. But it may never be executed due to delay in execution of a transaction [9]. In multi version transaction model [10], data is made available as soon as a transaction commits at a mobile host and another transaction can share this data. But data may be locked for a longer time at a mobile host before the lock is released at the database server.

In [11], AVI (Absolute Validity Interval) was introduced for enforcing concurrency control without locking. AVI is the valid life span of a data item. But it calculates AVI only based on previous update interval. In [12], a method based on PLP (Predicted Life Period), which takes care of the dynamicity of the life time of data was proposed. Here, life span of data is predicted based on the probability of updation of data item. This method makes PLP of data item very close to the actual valid life span of a data item.

In [13], a transaction model for supporting mobile collaborative works was proposed. This model makes use of Export-Import repository which is a mobile sharing work space for sharing data states and data status. But in the Export-Import repository based model, locking is the main technique which has the following disadvantages. (i) More bandwidth is needed for request and reply since the locking and unlocking requests have to be sent to the server. (ii) Disconnection of mobile host or a transaction failure will result in blocking of other transactions for a long period. Our framework is better than the model which uses Export-Import repository for sharing data since it minimizes message communication costs and data update costs to a larger extent. Also disconnected mobile hosts are treated separately within a mobile affiliation by waiting for their reconnection.

3. MOBILE AGENT BASED ARCHITECTURE

The proposed Mobile Agent based architecture model is illustrated in Fig 1. The model consists of Mobile Hosts,

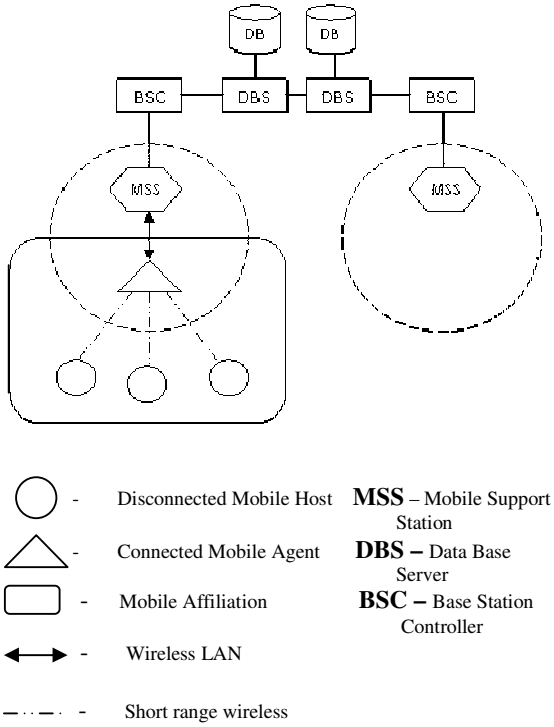

Fig.1. Mobile Agent Based Architecture Model

Mobile Support Stations (MSS), and Data Base Servers (DBS). MSS is connected with a Base Station Controller (BSC) [14] which coordinates the operations of MSS using its own stored program. Unrestricted mobility is supported by wireless link between MSS and Mobile Hosts. Each MSS serves one cell whose size depends on the power of its MSS. Data Base Servers are connected to the mobile system through wired lines as separate nodes. Each DBS can be reached by any MSS and new DBSs can be connected and old ones can be taken out from the network without affecting mobile communication. A DBS can communicate with a MH only via MSSs.

Mobile agent is a special mobile node which connects to the MSS to cache the frequently accessed data. Disconnected Mobile Hosts can connect to the Mobile Agent using short range wireless communication technologies to form mobile affiliation workgroup.

Mobile hosts are allowed to access data from the cache. When data request is made for the first time, data is retrieved from the server and stored in the cache. Subsequent requests are handled by the Data Access Manager module itself. When a mobile host requests for data update, after local updation of the data item, invalidation report is sent to all the mobile hosts that have already accessed the same data. This makes all the mobile hosts to refresh their data values. When a mobile host is disconnected from the Mobile agent after updation request, the updation task is transferred to the Data Access Manager in the Mobile Agent. Data Access Manager module is used to coordinate the operations in the cache.

After disconnected from the server, Mobile Agent can move along with the connected MHs and MHs can continue their

transaction execution. If data update at the server is requested, mobile agent will wait for reconnection before updation is made.

4. MOBILE TRANSACTION FRAMEWORK

When Mobile Hosts enter into the Mobile Agent area, they connect to it using short range wireless network technology to form Mobile Affiliation Work Group. Frequently accessed data are cached in the Mobile Agent. Mobile Hosts can access the cached data in the Mobile Agent. The Data Access Manager module at the Mobile Agent is responsible for enforcing concurrency and cache invalidation. Fig 2 and 3 illustrate the steps involved in MH, server and DAM (Mobile Agent) algorithms.

4.1 CONCURRENCY CONTROL MECHANISM

When multiple mobile hosts access data simultaneously the problem of data inconsistency arises. This problem can be solved if we use an efficient concurrency control mechanism. When data request is made for the first time, data is retrieved from the server and stored in the cache. Future requests for data are managed directly by the Data Access Manager.

Data Access Manager uses a suitable data item format to store data in the cache [12]. It has (id, TLU, PLP, dataval, NT) where id denotes unique Id of the data item, TLU indicates time of Last Update, PLP is Predicted Life Period, dataval is current value of the data item and NT denotes number of transactions that concurrently access the data item.

When Data Access Manager fetches data for the first time from the server, it sets TLU to current time, PLP to optimal time based on the nature of data item and NT to 1. NT is incremented whenever a new data access request is made. Data in the cache becomes invalid, once it is updated in the server. Life span of a data item is predicted using PLP. It makes use of the probability of updation as a basis for setting valid life span of a data item. In PLP interval, data item is valid and all the mobile hosts can access same data item concurrently.

When a MH makes update request or PLP expires, the data item is invalidated. Now PLP is modified and invalidation report is sent. The predicted life period of data item is computed using the formula given in [12].

$$PLP=PPLP \pm (p*PPLP)$$

where PPLP is Previous Predicted Life Period and p is predicted probability of updation of data item. p = Total_updates / NT. It is the ratio of data item update to data item access. Since predicted probability of updation is based on recent past history of updation rate, it is highly probable that PLP is very close to the actual validity interval of the data item.

4.2 TRANSACTION EXECUTION IN THE MH

After connecting to the server, MH gets a copy of a data item from the Data Access Manager using read request. If the MH wants to update data, it first checks whether it is a transaction update or not. If it is a transaction update, it will check if the MH is about to be disconnected. If so the updation task is assigned to the Data Access Manager before disconnection. Otherwise update request is sent to Data Access Manager.

4.3 FUNCTION OF DATA ACCESS MANAGER AND SERVER

When MH makes a read request, if it is in the cache of the Mobile Agent, NT is incremented by one. Otherwise, if Mobile Agent is connected to the server, it will fetch the data from the server and initialize data item format. If Mobile Agent is disconnected, the read request will be put in the queue and it will wait for Mobile Agent to get reconnection with the server. Once reconnection is got, it will fetch data from the server and initialize data item format.

When MH makes an update request, DAM updates data locally and invalidation report is sent to all the mobile hosts that have already accessed the same data item. This forces all the transactions to refresh their data values. In order to forward this update request to the server, it will check whether Mobile Agent is connected to the server. If so, update request is forwarded to the server. Otherwise, the update request will be put in the queue and it will wait until reconnection of Mobile Agent with the server is established. After reconnection with the server, update request is forwarded to the server. The server updates the data and sends invalidation confirmation along with the updated value. Once Data Access Manager receives the confirmation, it updates the data in the cache. The data in the cache is invalidated if updation is made in the server or PLP expires.

If transaction update is made by the Data Access Manager for the disconnected MH, the above procedure is followed except that at the end, DAM generates updation report and forwards it to the MH when it gets reconnected.

5. PERFORMANCE ANALYSIS

Simulation for the framework is done in Pentium Dual Core System @ 2.4 GHz with 3 GB RAM using Java 2. The results of the analysis are shown in Fig 4 and 5. Response time is calculated as the time taken to service the request made by the mobile host. We have compared the performance of our proposed model with the model proposed in [13]. Both the proposed and E-I models are simulated and the response time is computed.

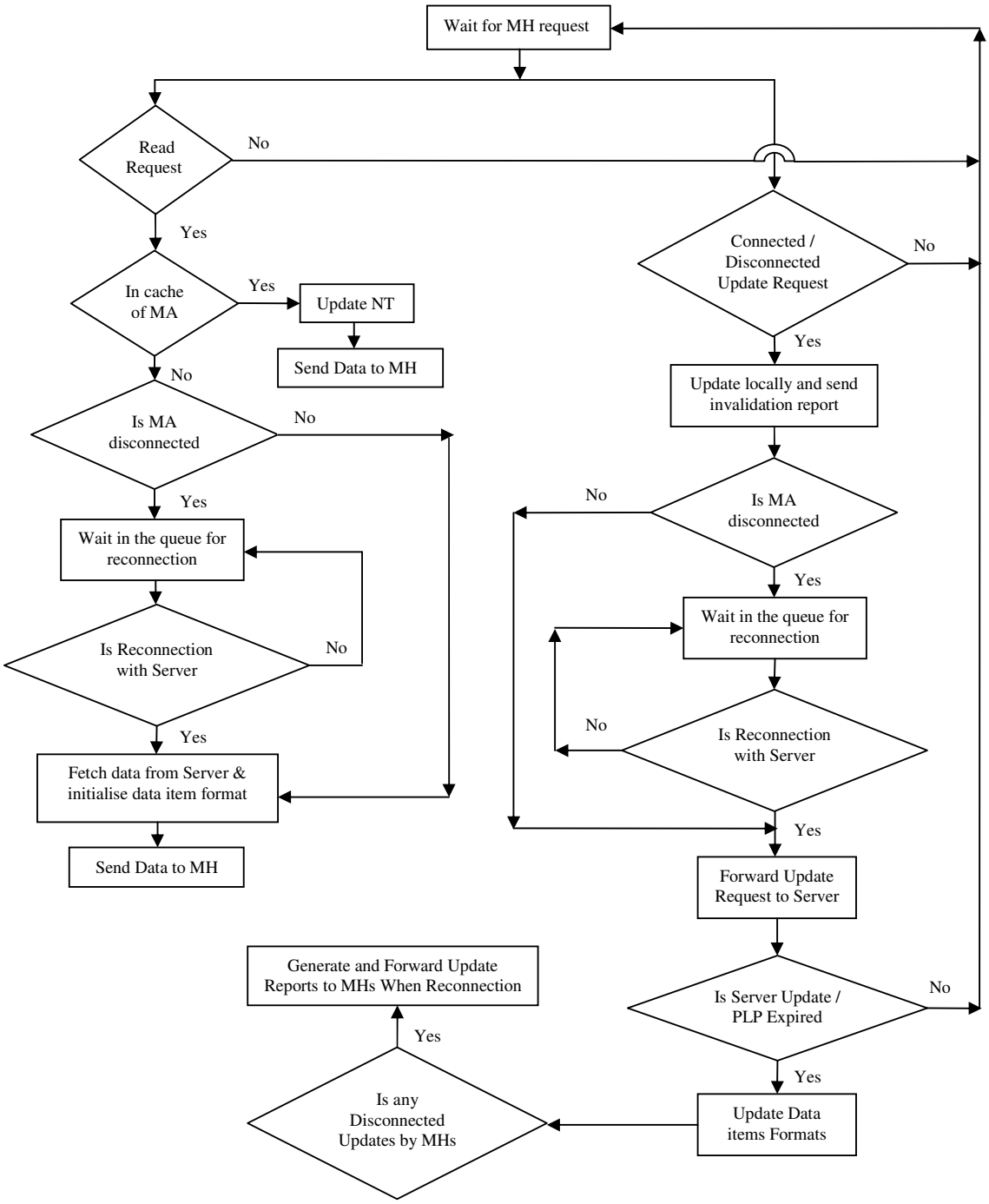

Fig.2. Flow Diagram for Data Access Manager

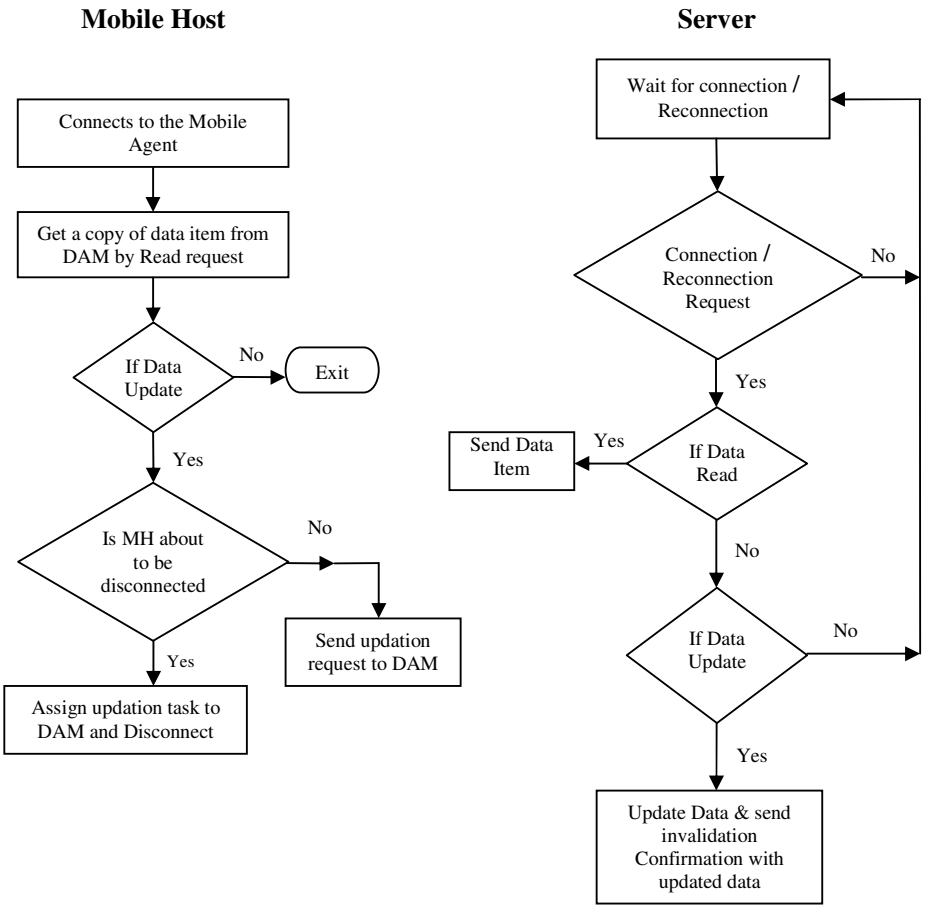

Fig.3. Flow Diagram for Mobile Host and Server

For transactions with Disconnection, when we compare response times of E-I repository model [13] with our model which uses Mobile Agent, E-I model takes more time. This is due to the extra time involved in communication overhead for locking mechanism of E-I repository model. The proposed model takes less time for transaction execution due to the cache invalidation technique used for enforcing concurrency control. The same is true for transactions without disconnect.

Moreover, the E-I model supports information sharing among transactions at different disconnected mobile hosts. Since physical export-import repository can be stored at one host or distributed among several hosts, the response time of E-I model is more. But in the proposed model, the data is accessed only from the cache of a mobile agent which results in less response time.

6. CONCLUSION

In this paper, we have proposed a transaction framework for disconnected mobile computing environment. Mobile Agent can form a Mobile Affiliation work group with the disconnected Mobile Hosts using short range wireless technology. The frequently accessed data are cached in the Mobile Agent. This cached data can be accessed by the mobile hosts when they get connected. When mobile hosts are disconnected from the Mobile Agent, transaction update task can be transferred to the Mobile Agent. By using a Mobile Agent and concurrency control without locking for accessing data, we claim that message communication costs and database update costs are minimized to a larger extent.

Table.1. Response time for Transactions without Disconnection

No. of Transactions	Response time in seconds	
	E-I Model	Mobile Agent Model
1	8.2	6.3
2	6.0	4.5
3	6.5	4.7
4	7.2	4.7
5	6.3	4.5
6	6.7	4.8
7	6.7	4.9
8	6.9	5.0
9	6.5	4.8
10	7.0	5.1

Table.2. Response time for Transactions with Disconnection

No. of Transactions	Response time in seconds	
	E-I Model	Mobile Agent Model
1	11.0	8.0
2	9.2	6.4
3	9.3	6.7
4	9.4	7.0
5	9.3	7.2
6	9.8	7.1
7	10.2	7.4
8	10.1	7.5
9	9.9	7.2
10	10.0	7.6

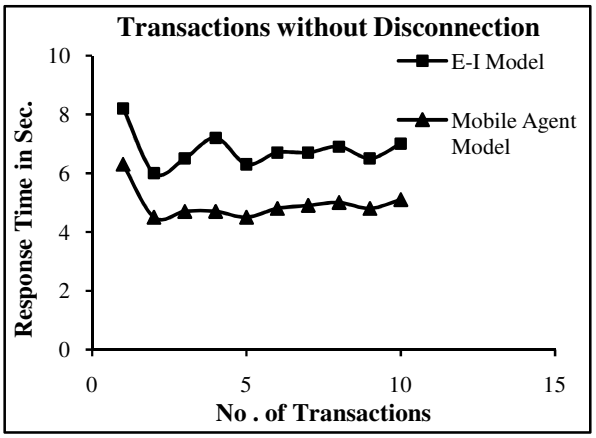

Fig.4. Analysis of Response time for Transactions without disconnection

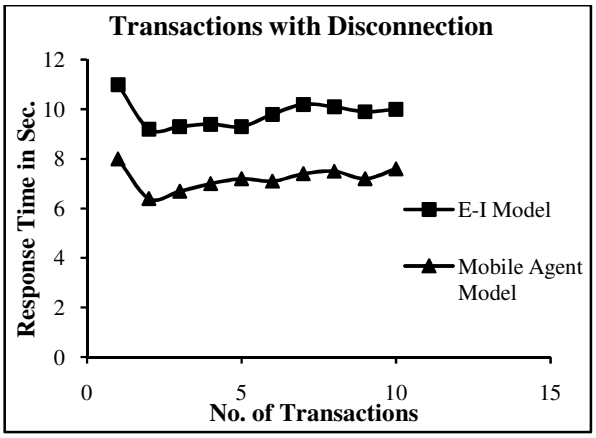

Fig.5. Analysis of Response time for Transactions with disconnection

REFERENCES

[1] T. Ozsu and Valduriez, 1999, "Principles of Distributed Database Systems", Englewood Cliffs, NJ: Prentice Hall.

[2] Borr, 1988, "High Performance SQL through Low Level System Integration," in Proc. ACM SIGMOD International Conference.

[3] Vijay Kumar, 2006, "Mobile Database Systems", Wiley Interscience.

[4] Victor C.S., Kwok-wa Lam and Sang H. Son, 2002, "Concurrency Control Using Timestamp Ordering in Broadcast Environments", The Computer Journal, Vol. 45, No. 4, pp. 410-422.

[5] Philip A. Bernstein, Vassos Hadzilacos and Nathan Goodman, 1987, "Concurrency control and Recovery in Database Systems", Addison – Wesley.

[6] V. Kumar, 1996, "Performance of concurrency control mechanisms in Database Systems". Englewood Cliff, NJ: Prentice Hall.

[7] D.J. Rosencrantz, R.E. Sterns, and P.M. Lewis, 1978, "System level concurrency control for distributed database systems", ACM Transaction of Database Systems, Vol. 3, No. 2.

[8] H.T. Kung and John T. Robinson, 1981, "On optimistic methods of concurrency control": ACM Transaction on Database Systems, Vol. 6, No. 2, pp.213-226.

[9] Ho-Jin Choi, Byeong-Soo Jeong, 2006, "A Timestamped Based Optimistic Concurrency Control for Handling Mobile Transactions", ICCSA, LNCS 3981, Vol. 3981/2006, pp. 796-805.

[10] Madria, S. K., M. Baseer, and S. S. Bhowmick, 2002, "A Multiversion Transaction Model to Improve Data Availability in Mobile Computing", CoopIS/DOA/ODBASE, pp. 322-338.

[11] Salman Abdul Moiz, Mohammed Khaja Nizamuddin, 2008, "Concurrency Control without Locking in Mobile Environments", International Conference on emerging trends in Engineering and Technology, pp. 1336-1339.

[12] Miraclin Joyce Pamila J.C and Thanuskodi K, 2009, "Framework for transaction management in mobile computing environment", ICGST-CNIR Journal, pp. 19-24.

[13] Le, H. N., and M. Nygård, 2007, "A transaction model for Supporting mobile Collaborative Works", International Symposium on Collaborative Technologies and Systems, pp. 347-355.

[14] Rajan Kurupillai, Mahi Dontamsetti and Fil J. Cosentino, 1997, "Wireless PCS: Personal Communication Services", New York: McGraw-Hill.