Commit Protocols in Mobile Environments: Design & Implementation

Salman Abdul Moiz[1], Dr. Lakshmi Rajamani[2] & Supriya N.Pal[3]

[1]Centre for Development of Advanced Computing, Bangalore, India
salman.abdul.moiz@ieee.org
[2]University College of Engineering, Osmania University, Hyderabad, India
drlakshmiraja@gmail.com
[3]Centre for Development of Advanced Computing, Bangalore, India
supriya@ncb.ernet.in

ABSTRACT

In any database environment either wired or wireless, if multiple host access similar data items it may lead to concurrent access anomalies. As disconnections and mobility are the common characteristics in mobile environment, preserving consistency in presence of concurrent access is a challenging issue. Most of the approaches use locking mechanisms to achieve concurrency control. This leads to increase in blocking and abort rate in mobile environments. However the dynamic timer adjustment strategies may use locking mechanism to efficiently implement concurrency control. To reduce deadlocks and blocking of resources an enhanced optimistic approach for concurrency control is proposed. To show the effectiveness of the commit protocols in mobile environments, a simulator is designed and implemented to demonstrate how the transactions are committed and how the data consistency is maintained when the transactions are executed concurrently. The simulator was tested for both pessimistic and optimistic approaches.

KEYWORDS

Concurrency Control, Coordinator, Participant, Mobile Host, Fixed Host, Transaction.

1. INTRODUCTION

Mobile computing is widely used in many applications such as mobile banking, traffic status, weather forecasting, etc., [10, 5]. In order to provide these services, required information is retrieved from database server via a wireless channel and is passed on to the mobile hosts.

Concurrency Control is one of the important components of transaction management. Several valuable attempts were made to efficiently implement the concurrency control strategies in mobile environment.

The concurrency control strategies presented in literature are based on three mechanisms viz., locking, timestamps and optimistic concurrency control. Due to various constraints in the mobile environment and nature of different online applications, these schemes may not work effectively. Several valuable attempts have been made to efficiently implement the commit protocols in traditional and mobile environments.

The choice of a protocol depends on type of the mobile application and the choice of the data service provider. In this paper the design and implementation of both protocols are presented.

The remaining part of this paper is organized as follows: Section 2 summarizes the survey of commit protocols proposed in the literature, section 3 describes the environment and the elements of mobile databases, section 4 specifies the conceptual architecture, section 5 describes the design & implementation of optimistic protocols, section 6 presents mobility issues, section 7 specifies the results and section 8 concludes the paper.

2. RELATED WORK

As long as the mobile clients are not involved in the concurrent access of data items, the database consistency can be preserved. When multiple mobile hosts initiate the transactions requesting for the same data item, it can be locked by only one of the transaction. After the execution of the transaction, the data items are unlocked and the same are acquired by the waiting transaction. When one transaction is being executed, the other transaction that needs the same data items, locked by the former has to wait for invariant time. The delay in acquiring the data items may further increase due to disconnections of mobile hosts for longer time. To solve these problems, following concurrency control techniques are proposed.

The basic idea is that a transaction has to be executed within certain time period (Execution time). This information is maintained by fixed host. To achieve concurrency control, two phase locking protocol was used in the traditional environment. However this protocol requires clients to communicate continuously with the server to obtain locks and detect the conflicts. Hence it is not suitable for mobile environments. In [3], A Timeout based Mobile Transaction Commitment Protocol uses timeouts to provide non-blocking protocol with restrained communication. It faces the problem of the time lag between local and global commit. In [4] the proposed Mobile 2PC protocol preserves the 2PC principle and minimizes the impact of unreliable wireless communication. This protocol assumes that all communicating partners are stationary hosts, equipped with sufficient computing resources and power supply with permanently available bandwidth.

In the pessimistic approaches, the items may be blocked for certain period of time. To avoid the blocking of data items and allowing multiple users to access the shared data items requires strong conflict resolution strategies. For this reason, an optimistic concurrency control technique is frequently used in wireless environments [13, 14, 15].

An optimistic concurrency control technique detects and resolves data conflicts in the phase of transaction validation. In a mobile environment the transaction validation is done on the server, it may lead to delayed response causing overhead at the server. An Optimistic Concurrency Control with Dynamic Time stamp Adjustment Protocol requires client side write operations. However because of the delay in execution of a transaction, it may never be executed [16]. In [17], the conventional optimistic concurrency control algorithm is enhanced with an early termination mechanism on conflicting transactions. However because of early termination a transaction need to be initiated again and again.
Optimistic concurrency control protocols (OCC) [19, 20, 22] are non-blocking and deadlock-free, which make them efficient to use in mobile computing and have been adopted in the Disconnected Operation [21] and Kangaroo Transaction model. However, without locks to data items, transactions might access conflicting data items under an optimistic concurrency control protocol (OCC). Two concurrent transactions conflict if one of them performs a write on similar

data items. Therefore, approaches to terminate conflicting transactions are proposed [17,18]. In these approaches if the conflict rate increases, more and more transactions get aborted.

Pessimistic commit protocols suitable for mobile environments are presented from [5] to [8]. Further a Real Time optimistic Commit protocols with a conflict resolution strategy is presented in [9]. The Design & Implementation of Pessimistic Commit protocols is described in [10]. The design and implementation of the optimistic commit protocol is presented in this chapter.

3. MOBILE DATABASE MODEL

3.1 Mobile Database Architecture

The mobile computing environment generally consists of three entities Fixed Host (FH), Mobile Hosts (MH) and Base Stations (BS) respectively. Terminals, desktop, servers are the fixed host, which are interconnected by means of a fixed network.

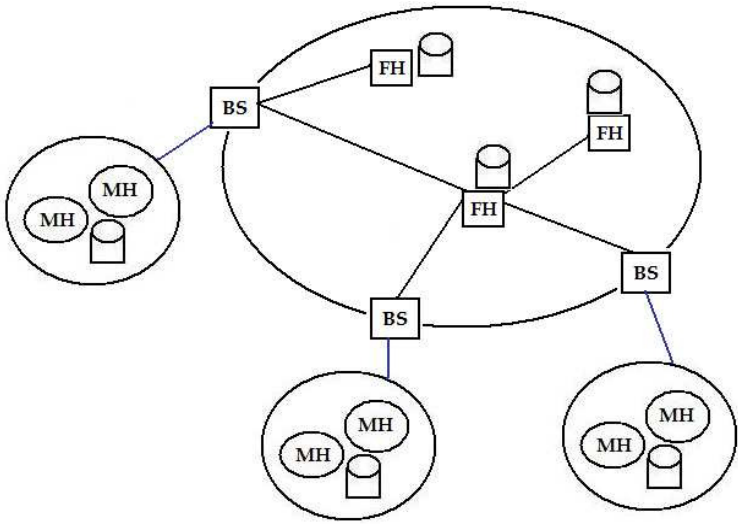

Figure 1. Mobile Database Architecture [9]

Large databases can run on servers that guarantee efficient processing and reliable storage of database. Fixed hosts perform the transaction and data management functions with the help of data base servers (DBS). Mobile units are the portable computers which can retain the network connections through the support of the Base Stations (BS).

The request for execution of a transaction is initiated at a mobile host may but it may be executed at fixed host or mobile host. A Base station connects to a mobile unit and is equipped with a wireless interface. It is also known as a Mobile Support Station. Mobile Hosts (MH) may not always be connected to the fixed network. They may be disconnected for different reasons. Mobile host may differ with respect to the computing power and storage space; however MH can run a DBMS module.

The commit protocols assume that the transaction request is sent to the Base Station which acts as the coordinator. The Mobile host and the Database systems act as the participants. The decision regarding the final commit, scheduling of transaction etc is done by the base station to which the mobile host was registered when it initiated the transaction request.

The component of the mobile transaction management system that are designed, implemented and tested includes the Pessimistic & Optimistic commit protocols.

3.2 Pessimistic Commit Protocols

The pessimistic approach says "No one can cause concurrency violation with my data if I don't let them at the data which I have it" [1]. In pessimistic commit protocols a row is locked for execution of the transaction. Once the transaction completes it is unlocked and can be acquired by the waiting transaction which needs the shared resource.
The pessimistic commit protocols uses locking of data items yet enhances throughput. Four variations of this pessimistic approach are presented. The Algorithmic approach uses static time out mechanisms to reduce the starvation of waiting process [8]. The Single Lock Manager Approach [5] uses the dynamic timer adjustment strategy to increase the overall commit rate. However to reduce the rollback operations the Concurrency control with reduced rollbacks [6] uses pre-emptive approach of scheduling the transactions and the strategy for concurrency control [7] uses predictive approach for scheduling the transaction execution. These approaches may be selected based on the business logic of the mobile application. The design and implementation of pessimistic commit protocols was presented in [10].

3.3 Optimistic Commit Protocols

In optimistic concurrency control strategy, the data items are not locked and can be used by more than one mobile host at the same time. The transaction executes in two phases: In the first phase the transaction is committed locally on mobile host using the on-demand approach[9]. In the second phase the results are updated onto the fixed host. In on demand multicasting , the data fragment needed by mobile host for executing the transactions is only requested from the mobile host and whenever the data conflict occurs, the invalidation reports along with request for re-execution of transactions is only sent to the mobile host using the shared data item.

In optimistic replication, shared data is replicated on mobile hosts and users are allowed to continue their work while disconnected. Updates performed by fixed host and later propagated to servers. In the earlier approaches whenever a concurrency violation occurs i.e data items are updated at fixed host the conflicting transaction using the similar data items was aborted. In this approach the conflicting transaction is not aborted but it is restated with new state of the data items.

4 ARCHITECTURE OF OPTIMISTIC CONCURRENCY CONTROL STRATEGY

The generic architecture of optimistic concurrency control strategy is represented in figure 2. This architecture is suitable for both online and offline transactions. However the emphasis is on offline transactions in this paper, where the transactions are executed at mobile host and later on the results are reconciled with the fixed host.

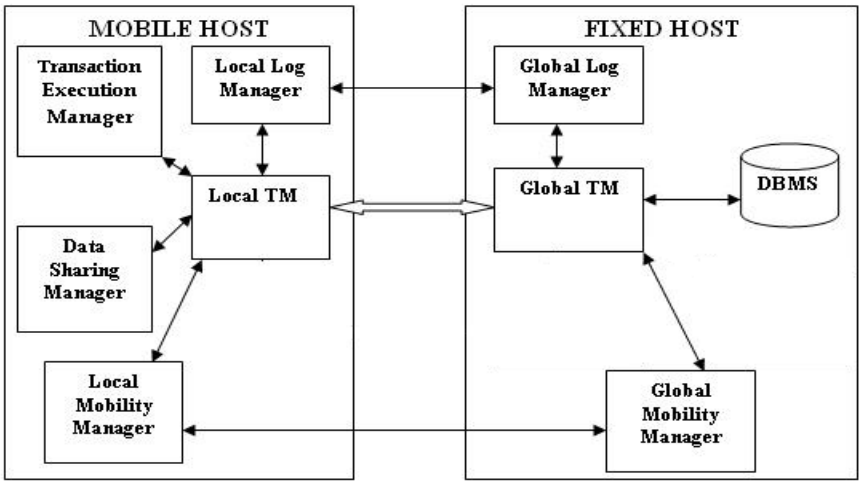

Figure 2. Architecture of Optimistic Concurrency Control Strategy

At Mobile Host, the *Local Transaction Manager* is responsible for managing the transactions at respective mobile hosts. When a local transaction is committed, the *Local Log Manager* maintains log records to support Durability. The commit record in the log storage helps in reconciliation of the results onto the fixed hosts. The *Data Sharing Manager* or cache manager manages the shared data which is obtained to initiate the offline transactions. *Transaction Execution Manager* is responsible for executing offline transactions and the *Local Mobility Manager* manages the handoff information when mobile host moves from one cell to another.

At Fixed Host, the *Global Transaction Manager* is responsible for managing the transactions submitted my mobile host after the offline processing. The *Global Log Manager* is responsible for making the final decision regarding the completion of a transaction and it is also responsible to maintaining durability after a failure. The *Global Mobility Manager* is responsible to manage sub-transactions to decide the final result of the transaction when the mobile host was on move during its execution.

The optimistic concurrency control supports offline transactions. Hence the mobile host initiates the request for the transaction. The Base station acts as a Global Transaction Manager.

The base station acts as the coordinator and the Mobile hosts and the Database System as participants. In figure 2 the architectural elements specified at fixed host i.e. Global Transaction Manager, Global Log Manager & Global Mobility Manager are logical components of the coordinator i.e. the base station.

5 DESIGN & IMPLEMENTATION OF OPTIMISTIC COMMIT PROTOCOLS

In the optimistic approaches, the data item requested for offline transactions are not locked, rather the transaction proceeds tolerating the conflicts. Later at the time of reconciliation of results the conflict is detected and respective resolution strategy is adopted.

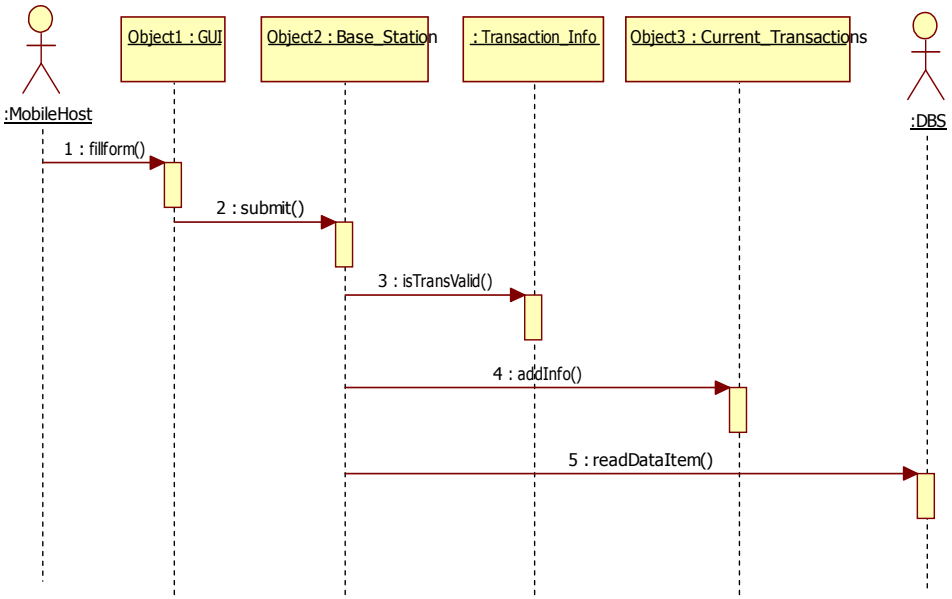

Figure 3. Behaviour of Initiating a Transaction

Figure 3 depicts the behaviour of Initiate Transactions for Optimistic strategy for concurrency control. The implementation of optimistic protocols is divided into two phases. In the tentative phase, the data items are read by mobile host and the transaction is executed locally. In the commit phase, the final commit decision is made by resolving conflicts.

In offline transactions, whenever a mobile host initiates the transaction (1: fillform()), the request is sent to the base station (2: submit()). The base station checks whether the requested transaction can be executed (3: isTransValid()), by comparing the Transaction_id with the one which is requested by the mobile host. Then the information regarding the new transaction request is entered in Current_Transaction relation (4: addInfo()). The data items are then read by the mobile host (5: readDataItem()). Figure 5 describes the behavior of executing a transaction.

When the transactions are committed locally (1:localCommit()), the local transaction manager sends the result (2: Propagate()) to the coordinator. The Global Transaction Manager at the Base station makes a decision regarding committing the transaction (3: isCommit()) based on the forward or backward validation strategy.

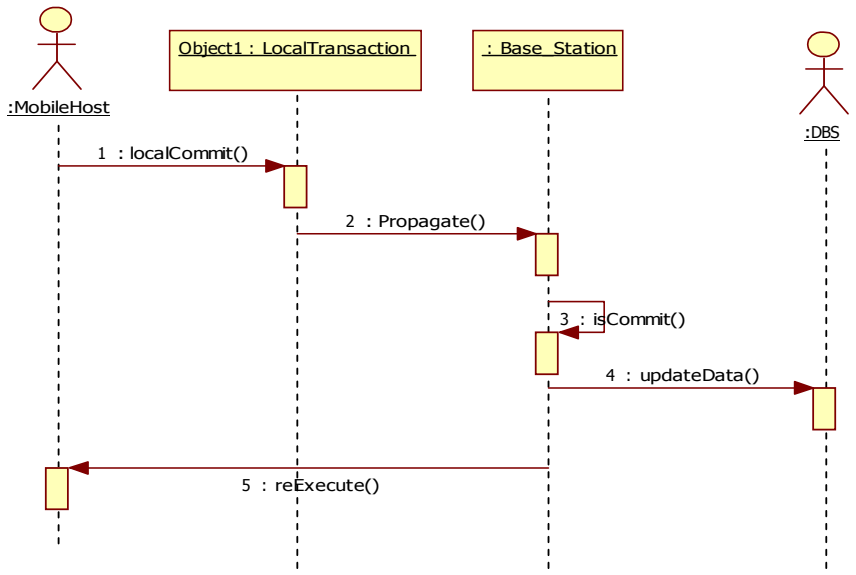

Figure 4. Behaviour of Executing A Transaction

The result is then updated to the database (4: updateData()). If more than one mobile host were sharing the same data items, then there exists a conflict. Conflict is resolved by multicasting the updated values to those mobile host which are the vicitims of data conflict. The new value of the data is sent to the mobile host by instructions to restart the execution of the transaction (5: reExecute()) at the respective mobile host.

6. MOBILITY MANAGEMENT IN OPTIMISTIC STRATEGY FOR CONCURRENCY CONTROL

Whenever a mobile host moves from one cell to another during the execution of a transaction, it registers with the mobile host in foreign cell. Figure 5 describes the behavior of execute transaction for optimistic protocols when the mobile host moves from one cell to another

When a mobile host completes the transaction in foreign cell (1: localCommit()), it sends the result to the base station to which it is registered to (2: propagate()) . The base station checks whether the mobile host started its execution in the same cell (3: isMhInHloc()) i.e in home location. Otherwise the result is propagated to the base station where the transaction started the execution (4: Propagate ()).

The base station makes a final commit decision (5: isCommit()). Once the coordinator decides to commit the transactions the results are updated in the database (6: updateData()). In case of any conflicts with other mobile host, the new values of data items are propogated to the mobile host using the conflicting data items (7: reExecute()).

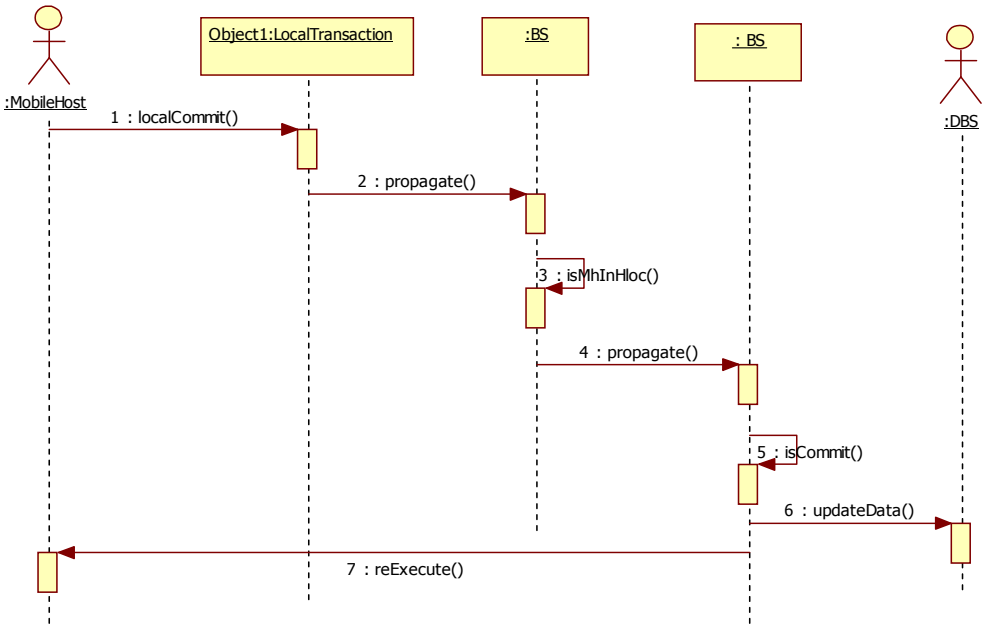

Figure 5 Handoff During Transaction Execution

7. RESULTS

In this section we compare the results of the proposed Real Time Optimistic Concurrency Control strategy with the optimistic strategies proposed in the literature.

In OCC/DTA (Optimistic Concurrency Control-Dynamic Timer Adjustment) [23], client side validation procedure is used that enhances the transaction throughput by adjusting the serialization order. In validation phase the mobile clients perform partial validation for committing a transaction. If the partial validation is successful global validation is performed. A transaction is expected to complete its execution within the specified time stamp. If the Lower bound crosses the higher bound value the transaction is aborted. As the mobile hosts have varying processing capabilities, restricting a transaction to be executed within certain timestamp may increase the abort rate. When a transaction is aborted, it may have to request for execution later, this unnecessary increase the uplink bandwidth too.

As proposed in the real time optimistic strategy the when a conflict is detected the transaction may not initiate the transaction request again, instead the updated value of conflicting data items if any will be multicasted to those mobile hosts using the shared data item with a request to re-execute the transaction. Though there may be possibility of increase in re-execution request, but this reduces the abort rate, decreases the uplink bandwidth and also avoids sending irrelevant information to the mobile host by using multicasting.

FBOCC [15] adopts optimistic concurrency control technique suitable for broadcast environments. In this technique the transaction is aborted immediately upon detection of conflict without adjustment of any serialization order. When the transactions are aborted immediately after detecting the conflicts, the abort rate is much higher and uplink bandwidth will almost be equivalent to the downlink bandwidth.

The proposed optimistic concurrency control strategy is compared with FBOCC. To simulate the results the Mobile transactions were generated at each client to measure the number of aborted transactions due to data conflict. As the number of mobile clients gets increased, the frequency of data conflicts gets increased. The maximum numbers of clients used to generate

different transactions were 20. At any instant of time 50% of transactions used the same shared data items.

Table 1. Comparison of FBOCC with the Proposed Optimistic Strategy

No. of Clients	Abort rate		No. of times uplink bandwidth is utilized	
	FBOCC	Proposed Strategy	FBOCC	Proposed Startegy
2	1	0	2	2
4	3	0	3	0
8	7	1	7	1
12	11	2	11	2
16	15	3	15	3
20	19	4	19	4

In FBOCC the transaction is aborted once a conflict is detected, whereas in proposed protocol, the transaction is re-executed with new values of data items. However it might be aborted after 3 attempts (say) of re-execution requests. The number of re-execution requests varies from one transaction to another and/or one application to another. The number of possible attempts for re-execution of a transaction is maintained at the fixed host.
The results in table 7.3 are based on the speed of execution of a transaction & the execution request was sent to the coordinator dynamically. A mobile host might request for execution quiet later but can be committed earlier. In the first case 2 clients have requested for same shared data item. In the second case 4 clients have requested for same shared data items. There is a possibility the transaction which was aborted might have also requested for execution of transaction immediately. It is also obvious that as the abort rate increases the uplink bandwidth also increases.

8 CONCLUSION

The Optimistic Concurrency Control Strategy doesn't use any locking which doesn't block the shared resources. Further concurrency can be guaranteed by first executing transactions locally and later on propagating the results. In this scheme whenever a fixed host detects a concurrency violation, it propagates the updated shared data item to the mobile host using the same data item without aborting it. The mobile host which successfully completes the transaction locally will be committed irrespective of its arrival time. In this scheme there could be a possibility that the transaction which arrived quiet early might not get executed because the other mobile hosts are executing faster. The future work may introduce a priority field to give chance to the transaction which requested first or a hybrid approach for concurrency control that enters into pessimistic approach by partially locking data items to complete its execution is needed.

REFERENCES

[1] Smith Wayne Plourede: "*Handling Concurrency Issues in .NET*", http://www.15seconds.com/issue/030604.htm, Web retreieve on February 19, 2005).

[2] Samidip Basu & Syed M. Rahman, "*Improving Optimistic Concurrency Control using Hybrid Techniques of Snapshot Isolation & ROCC*", Proceedings of Mid-west Instruction & Computing Symposium, 2006.

[3] Vijay Kumar, Nitin Prabhu, Maggie Dunham, Ayse Yasemin Seydim, "*TCOT - A Timeout based Mobile Transaction Commitment Protocol*", IIS 9979453, 2004.

[4] Nadia Nouali, Anne Doucet, Habiba Drias, "*A Two-Phase Commit* Protocol *for Mobile Wireless Environment*", Vol. 39, 16th Australasian Database Conference, 2005.

[5] Salman Abdul Moiz, Dr. Lakshmi Rajamani, "*Single Lock Manager Approach for achieving Concurrency in Mobile Environments*", 14th IEEE International Conference on High Performance Computing (HiPC), 2007. Springer LNCS 4873, ISBN 978-3-540-77219-4, pp. 650-660, 2007.

[6] Salman Abdul Moiz, Dr. Lakshmi Rajamani, "*Concurrency Control Strategy to Reduce frequent rollbacks in Mobile Environments*", 2009 IEEE/IFIP International Symposium on Trusted Computing (TrustCom 2009), ISBN# 978-0-7695-3823-5, Vol 2. Pp. 709-714, 2009.

[7] Salman Abdul Moiz, Dr. Lakshmi Rajamani, "*An Efficient Strategy for achieving Concurrency Control in Mobile Environments*", 12th IEEE Asia Pacific Network Operations & Management (APNOMS) symposium, 2009. Springer LNCS 5787, ISBN# 978-3-642-04491-5, pp. 519-5222, 2009.

[8] Salman Abdul Moiz, Dr. Lakshmi Rajamani, "*An Algorithmic approach for achieving concurrency in Mobile Environments*", 1st National Conference on Computing for Nation Development, INDIACom 2007, ISBN #978-81-094526-0-1, ISSN # 0973-7529.

[9] Salman Abdul Moiz, Dr. Lakshmi Rajamani, "*A Real Time Optimistic Strategy to achieve Concurrency control in Mobile Environments using on-demand multicasting*", International Journal of Wireless & Mobile Networks (IJWMN), Vol.2, No.2, May 2010, pp. 172-185, ISSN #0975-3834 (online), ISSN #0975-4679 (print).

[10] Salman Abdul Moiz, Dr. Lakshmi Rajamani, Supriya N. Pal, " *Design and Implementation of Pessimistic Commit Protocols in Mobile Environments*", The First International Workshop on Database Systems (DMS-2010), Springer Verlag Berlin Hiedelberg, CNSA 2010, CCIS 89, pp. 603-612, 2010

[11] Barbara, D., "Mobile computing and databases-a survey", IEEE Transaction of Knowledge Data Engineering, volume 11 issue 1, pp.108-117, 1999

[12] Imielinski, T. and Badrinath, B. R., "Mobile wireless computing: challenges in data management", Communications of the ACM 37(10), pp. 18-28, 1994

[13] Mei-Wai Au, Edward Chan and Kam-Yiu Lam, "Concurrency Control for Mobile Systems with Data Broadcast", Journal of Interconnection Networks, pp.253-267, 2001

[14] Pitoura E., "Supporting Read-Only Transactions in Wireless Broadcasting", Proc. DEXA Workshop on Mobility in Database and Distributed Systems, pp.428-433, 1998.

[15] Victor C.S.Lee, Kwok Wa Lam, Tei-wei Kuo,"Efficient Validation of Mobile Transactions in Wireless Environments", The Journal of Systems and Software 69(2004), 183-193.

[16] Ho Chin Choi, Byeong-Soo Jeong, , "A Timestamp-Based Optimistic Concurrency Control for Handling Mobile Transactions, Springer Verlag, LNCS 3981, PP. 796-805, 1996

[17] Anand Yendluri, Wen-Chi Hou, and Chih-Fang Wang, "Improving Concurrency Control in Mobile Databases", Springer Verlag LNCS 2973, PP. 642-655, 2004.

[18] Barbara D. and T. Imielinski. "Sleepers and Workaholics: Strategies in Mobile Environments," Proc. ACM pp. 1-12, May 1994.

[19] Bernstein, P.A, Hadzilacos, V. and Goodman, N, "Concurrency Control and Recovery in database System", Addison-Wesley 1987

[20] H. T. Kung and J. T. Robinson, "On Optimistic Methods for Concurrency Control," ACM TODS, 6(2), June 1981.

[21] J. Kisler, and M. Satyanarayanan, Disconnected Operation in the Coda File System, ACM Transactions on Computer Systems, 10(1), 1992.

[22] T. Härder. "Observations on optimistic concurrency control schemes". Information Systems, 9(2):111–120, 1984.

[23] Ho Chin Choi, Byeong-Soo Jeong, "*A Timestamp-Based Optimistic Concurrency Control for Handling Mobile Transactions*", Springer Verlag, LNCS 3981, PP. 796-805, 2006.

Authors

Salman Abdul Moiz is a Research Scientist at Centre for Development of Advanced Computing, Bangalore. He received his B.Sc from Osmania University, MCA from Osmania University, M.Tech(cse) from Osmania University and M.Phil(Cs) from Madurai Kamaraj University.

He is a Research Scholar at Osmania University and published 25 papers in various National/International Conferences and Journals. His areas of interests include Mobile databases, Software Process Improvements, Agile Methodology & Disaster Recovery.

Dr. Lakshmi Rajamani is working as Professor & Head of the Department, CSE, University College of Engineering, Osmania University, Hyderabad. She received M.Sc (Statistics) from IIT Kanpur, M.Phil (Computer methods) from University of Hyderabad and PhD (CSE) from Jadavpur University, Kolkata.She authored more than 30 papers in various National/International conferences and Journals. Her research interests are in the areas of Neural Networks, Artificial Intelligence, Distributed Computing & Data Mining.

Supriya N. Pal holds a Master's degree in Computer Science from the University of Mumbai. She is a Technical Lead in applied research projects of the Database Systems & Software Engineering division in C-DAC, Electronics City, Bangalore. Her research interests include SW Re-engineering, SOA, Messaging middleware, and Mobile Computing and its applications.