CONCURRENCY CONTROL IN MOBILE ENVIRONMENTS: ISSUES & CHALLENGES

Salman Abdul Moiz[1], Supriya N.Pal[2], Jitendra Kumar3, Lavanya P[4], Deepak Chandra Joshi[5], Venkataswamy G[6]

[1]Research Scientist, Centre for Development of Advanced Computing, Bangalore
Salman.abdul.moiz@ieee.org
[2]Associate Director, Centre for Development of Advanced Computing, Bangalore
supriya@cdac.in
[3]Senior Staff Scientist, Centre for Development of Advanced Computing, Bangalore
jitendra@cdac.in
[4, 5, 6] Staff Scientist, Centre for Development of Advanced Computing, Bangalore
(lavanya, deepak, venkataswamy)@cdac.in

ABSTRACT

The use of data services from handheld devices has increased exponentially resulting in several challenges. Transactions requiring the same shared data item may simultaneously perform a write operation leading to inconsistency of data items. The generic characteristics of mobile environments like variable bandwidth, disconnections, mobility etc makes transaction management more difficult thereby the traditional Isolation property may not be guaranteed. This paper analyzes and compares various concurrency control strategies in mobile environments proposed in literature. The design requirement for preserving isolation property in mobile environments is also presented.

KEYWORDS

Concurrency Control, Pessimistic Strategy, Optimistic Strategy, Mobile Host, Fixed Host, ACID.

1. INTRODUCTION

Transactions are key to structuring distributed applications. Concurrency control is one of the important building blocks of transaction management. The requirement of concurrency control arose to ensure correctness when shared data item is updated by multiple transactions at the same time. The traditional two phase locking protocol is not suitable in mobile database environment due to frequent mobility and disconnections.

Further to manage the data correctly, the support of ACID (Atomicity, Consistency, Isolation & Durability) properties of transaction should be revisited with respect to the characteristics of mobile database environment. Atomicity is guaranteed by the commit protocols, Consistency is assured by DBMS or application programs if it can ensure certain specified constraints, Isolation is guaranteed by concurrency control manager and Durability is ensured by logging.

Transaction management is characterized by the environment in which transaction gets executed. In the first group the transaction are completely or partially executed at mobile host. In this group the focus is on support of ACID properties. In the second group, transactions initiated at mobile hosts are executed at fixed host. In this group ACID properties are not compromised and the focus is on support of executing transactions successfully in presence of mobility and disconnections [1].

The remaining part of this paper is organized as follows: Section -2 describes the architecture of mobile database system, execution modes in mobile environments and the need of mobile middleware and its architecture. Section -3 presents the concurrent access anomalies and the lists the challenges in achieving Isolation property in mobile environments. Section-4 presents the pessimistic strategies for concurrency control in mobile environments and along with issues and challenges. Section -5 describes the optimistic strategies in mobile environments followed by issues and challenges and Section -6 concludes the paper.

2. MOBILE DATABASE ENVIRONMENT

The following figure describes the reference architecture of Mobile Database System. It resembles C&C (Component & Connector) view style architecture.

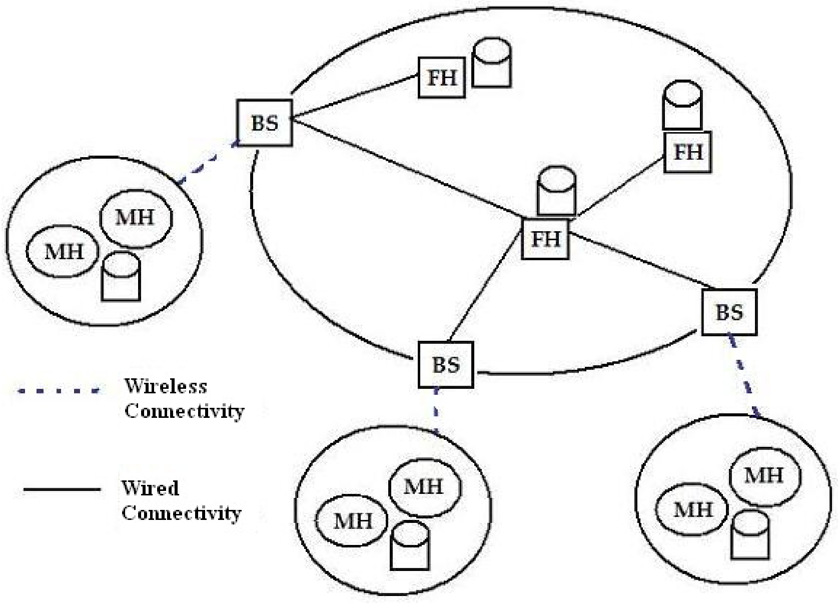

Figure 1. Reference architecture of Mobile Database System

The Mobile database system consists of two major components and two major connectors. The two components are Mobile Host and Fixed Host respectively and the two connectors are Wireless and Wired Networks.

Terminals, desktops, servers are the Fixed Host which are interconnected by means of a fixed network. Large databases can run on servers that guarantee efficient processing and reliable storage of database [1].

Mobile Hosts (MH) like Palmtops, Laptops, PDA's or Cellular phones is not always connected to the fixed network [1]. They may be disconnected for different reasons. Mobile host may differ with respect to the computing power and storage space; however MH can perform database transactions. The disconnections may be soft or hard. Hence disconnections are handled as normal situations and not as failures.

Advances in wireless technology and ubiquitous computing have created a new domain called mobile computing, where the users access the data irrespective of their physical location and movement behavior. The wireless technology provides users to logically retain the network connection even when there is a handoff i.e. when the mobile host is moving from one location to another. The geographical area covered by a base station is called a cell. Every time the mobile host communicates with the base station of the cell it belongs to. The process of entering new cell is said to be a handoff or handover.

2.1. Execution Modes

The transactions are initiated at mobile host but may be executed on mobile host or fixed host or the execution may be distributed between mobile host & fixed host [1] respectively.

- Complete Execution on the Fixed Network

In this execution mode, the transaction is initiated at mobile host but is completed executed at fixed network. In this model, the mobile host acts as a thin client.

- Complete Execution on a MH

In this mode, transactions are initiated at mobile host and are executed on mobile host. This approach requires mobile hosts to have processing & storage capabilities as well as managing data. However reconciliation is needed with the fixed host at some point in time.

- Distributed Execution on a MH and the Wired Network

In this mode, transactions are initiated at mobile hosts and the execution is distributed among mobile host and fixed host. A sub-transaction is executed at mobile host and another one at fixed host. This helps in minimizing the communication between the fixed host and mobile hosts respectively.

- Distributed Execution among several MHs

In this mode, the transaction is distributed among several mobile hosts for execution. It provides a peer-peer strategy. A mobile host acts as a server for other mobile hosts so that the execution is distributed between them. The selection of a mobile host for execution of a transaction is location based.

- Distributed Execution among MHs and FHs

This mode provides a fully distributed environment where a transaction execution is distributed among several mobile hosts and fixed hosts respectively. In this transaction execution, multiple parties may be involved. For example a customer who wishes to purchase products may connect to the supermarket and as well as bank and finally the results are to be reconciled successfully on individual fixed hosts.

2.2. Mobile Middleware

Mobile Middleware is software that acts as an interface between the Mobile host and wired fixed host. The functions of Mobile Middleware are

- It must deliver the information precisely to the right place irrespective of mobility, disconnections, interface constraints etc.,
- It has to enable deployment of a standardized mobile application on mobile devices with varied operating systems, screen sizes etc.
- Need to secure data on devices rather than just securing data communications

Figure 2 depicts the typical mobile middleware architecture. It resembles three-tier architecture and has to support cross platform characteristics. This is because the mobile devices works on different platforms and operating systems and the applications need to be accessed from any of the devices.

Mobile middleware communicates with the fixed host on behalf of the mobile hosts. The mobile middleware also helps in providing location transparency thereby reducing the overhead of hard coding the applications on mobile devices.

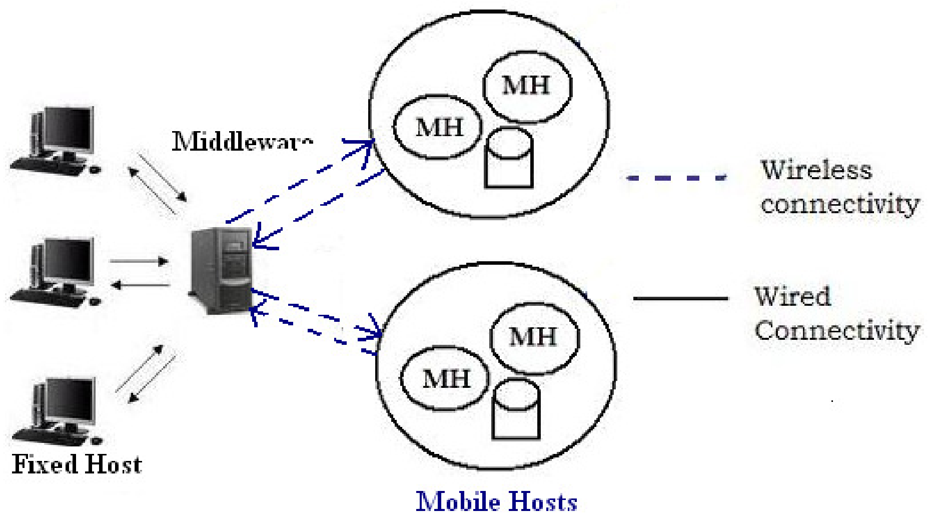

Figure 2. Mobile Middleware Architecture

The mobile middleware helps in managing transactions, providing middleware security and support for cross platform implementation. The Cross platform middleware helps in supporting wide range of mobile applications.

It also provides Device flexibility & Application flexibility. The transaction management in mobile environments can be implemented on the mobile middleware. The data synchronization strategies can be realized by mobile middleware.

3. CONCURRENT ACCESS ANOMALIES IN MOBILE ENVIRONMENTS

When multiple mobile hosts request for same data items, it may lead to concurrent access anomalies. To preserve Isolation property pessimistic and optimistic strategies are proposed in literature.

In a pessimistic strategy, the traditional database uses locking protocol to overcome the problems of concurrent access. These protocols are not suitable for preserving consistency in presence of concurrent access of mobile hosts. A transaction initiated at a mobile host locks the required data items. If another mobile host requires the same data item it needs to wait till the mobile that requested the resource first unlocks the data item. If the mobile host doesn't commit the transaction, the second mobile host waiting in a queue has to wait for invariant time. This leads to starvation. Further as the disconnections are treated as normal behaviour in mobile environments, developing an efficient concurrency control strategy is a challenging problem.
In an optimistic strategy, whenever multiple sites request for same data items, they are allowed to read the data item thus tolerating conflicts to occur. However at the time of global commit, conflict resolution strategy is applied to get back the system into consistent state. In mobile

database environment, the global commit operation may increase the uplink bandwidth as the mobile host has to initiate the transactions again and again.

4. PESSIMISTIC STRATEGIES IN MOBILE ENVIRONMENT

Pessimistic strategy works on the following principle

"Don't allow any transaction to use the same shared data item, unless it is completed".

In Pessimistic strategy, the data items needed for executing a transaction are locked. Once the transaction is successfully completed, it is unlocked then the data items can be used by the transaction waiting in queue.

Since disconnections are normal in mobile environments, a mobile host may lock the data item and may not release the locks invariantly. To overcome the problem of starvation, each transaction is associated with a timer. If the results of the offline transactions are not reconciled within a stipulated timer period, the transaction may be aborted and the data items are acquired by waiting transaction.

Pessimistic strategies such as Two Phase Locking (2PL) require frequent message exchanges with fixed hosts [2]. This reduces the throughput of the system. Further unpredictable disconnections may also lead to undefined locking time.

4.1 Distributed High Priority Two Phase Locking (DHP-2L) Protocol

DHP-2L [3] is a distributed real time locking protocol based on high priority two phase locking (HP-2PL) to achieve concurrency control in mobile environments. Conventional concurrency protocols schedule transactions on equal basis. However, the HP-2PL [4] restarts a lower priority transaction if a higher priority transaction wants to set a lock which is held by higher priority transaction.

A transaction is said to be local if it access data items available at only one base station. Otherwise it is a global transaction. DHP-2PL is a blocking protocol that resolves lock conflicts based on the priority of the transactions. Let T_r represents the lock requesting transaction and T_h the lock holding transaction, the following table specified the process of resolving lock conflicts in DHP-2PL.

Table 1. Lock Conflict resolution in DHP-2PL [3]

```
 Lock Conflict (Tr, Th)
 Begin
   If Priority(Tr) > Priority(Th)
        If Th is not committing
              If Th is a local transaction
                   Restart Th locally

Else
                   Restart Th globally
             End if

 Else
             Block Tr until Th releases the lock
             Priority(Th):= Priority(Tr) + fixed priority level
```

```
     Endif

Else
        Block $\mathbf{T_r}$ until $\mathbf{T_h}$ releases the lock
    End if

     End
```

However DHP-2PL uses transaction restart mechanism to resolve lock conflicts between non-committing transactions. Further Global restarts takes more time than the local restart operations.

4.2 Timeout based Mobile Transaction Commit Protocol (TCOT)

TCOT [20] works on timeout based principle to reach the final state of the transaction execution by avoiding starvation of the shared data items. It works in an offline mode where the fragment needed for executing a transaction is read onto the mobile client and later the results are integrated with the server

.

If the transaction can’t be executed within the time period “t”, the timer value can be increased by sending the request again to the fixed host. This increases the uplink bandwidth. Further at the time of request of change in timer value, the mobile host may face the connectivity issues and it faces the issue of time lag between local and global commit operation.

4.3 Other pessimistic strategies

In [21] Mobile speculative locking protocol is introduced to reduce the blocking of transaction if two phase locking is employed. This approach requires extra resources at the mobile host to carry out speculative execution. In [22], a concurrency control scheme is proposed which in which the priority based scheme is used to serialize the transaction. Whenever a conflict occurs the priority is given to the older transaction. This is similar to the traditional locking and if the priority is given to the mobile host which cant execute the transaction for a long time, it not only suffers from starvation but the throughput decreases considerably. In [23], a concurrency control strategy is presented which sets the validity of data items based on the number of data items required. As the data items needed to execute a transaction might be distributed over several database servers, there is a need for a middleware which could handle such situations.

4.4 Issues & Challenges

Pessimistic protocols implemented in traditional environments are not suitable to mobile environments due to the inherent characteristics like mobility and Disconnections.

Timeout based commit protocols are needed in mobile environments to avoid starvation problems as a result of indefinite locking of data items by a particular mobile host.

The timer value may vary from one transaction to another. However if the transaction is aborted after expiry of timer, the transaction is to be requested again. This increases uplink bandwidth. To overcome this drawback if the transaction is not executed within specified time period it may be rolled back and may be executed again if no other transaction is waiting for same data items in the queue. Otherwise round-robin approach is followed. The qualitative analysis of the pessimistic strategies for guaranteeing concurrency control in mobile environments is presented in table 2.

Table 2. Qualitative analysis of Pessimistic Strategies for Concurrency Control

Pessimistic Strategy	Transaction Type	Waiting Time	Transaction throughput	Transaction Aborts	Transaction restarts
DHP-2L	Flat	Depends on transaction priority	Depends on priority of inheritance	Depends on firm deadline	More local/global restarts
TCOT	Nested	Depends on max time of execution	Decreases when there are frequent request for increase in timer value	Depends on threshold value	Transaction restarts if the execution is not completed in threshold limit
Mobile Speculative Locking	Nested	Depends on time taken by nested transaction	Less as more memory & computation capability is needed	Comparatively high as for every abort of nested transaction, an abort occurs	The intra and inter transaction parallelism may lead to high restarts due to disconnection s
Prioritized Concurrency Control scheme	Flat	Depends on time taken by high priority transaction	May decrease when the disconnection time is higher	Depends on current access time and data validity period	May reduce but execution time increases
Data count driven concurrency control scheme	Flat	Based on number of required data items	May reduce if transaction requires more data items and sever is heavily loaded	Depends on validity period and number of data items needed	May reduce but time taken for lookup of data items and counting them will be high

5. OPTIMISTIC STRATEGIES IN MOBILE ENVIRONMENT

Optimistic strategy works on the following principle:

"First perform operations, check for conflicts later".

In Optimistic Commit Protocols, conflict is tolerated in read phase. However to get back the system into a consistent state conflict resolution technique is applied at validation phase.

In this protocol multiple users are allowed to read the same shared data item simultaneously. When the transaction commits, the data items read by other users have to be invalidated. To implement this caching techniques are presented in literature which uses broadcasting i.e. whenever a commit is done locally, an invalidation report is broadcasted to all clients to invalidate the data items.

5.1 The Concurrency Control Mechanism (CCM)

In CCM [5], a modified form of two tier replication scheme is proposed to reduce the no of transaction rejections and transaction commit time at the mobile hosts. In this approach the number of replicas allowed for shared data items are finite. If the possible number of replicas is set to "n", only "n" mobile hosts can locally store a copy of data item for execution. The validity of a data item is control by timeout parameter. If the data item is not updated or the results are not returned within specified period of time, it can't commit the transaction. The timeout values are assumed to be multiple of broadcast cycle time. If the transaction is not committed within specified time, it is blocked and has to wait till a new value of timer is assigned to the mobile host.

It assumes that the value of timeout should be sufficiently large so that the updates can be propagated to the fixed host. However this may reduce the throughput of the system. Further in CCM if the transaction is committed within the specified time period it can update the data on fixed host without waiting for the result of base station. This increases the load on mobile host as it has to broadcast the status of data item to other mobile hosts which holds the same replica.

Further in this approach after the expiry of timer value, DBS always has to send the new values of number of replicas and time for validity of data. Further if the number of applications deployed on a mobile increases, then the cache should also increase to maintain the data item information.

5.2 Sequential Order with Dynamic Adjustment (SODA) Strategy

SODA [6] guarantees consistency of concurrent transactions in mobile P2P databases. It reduces response time by applying the concept of sequential order and reduces the abort rate by dynamically adjusting the sequential order.

SODA is a three phase commit protocol. In *Read & Compute phase*, transaction T reads the set of data items assigns a timestamp and stores them locally. T also computes the new values (write set) and maintains it locally. In *Validation Phase*, the read and write set are validated against set of committed transactions. If the transaction T passes the validation phase a new time stamp is assigned which is used as the commit time of transaction T. In Commit and Write Phase, if the transaction succeeds the validation phase, it can write values of write set into database. Otherwise it may be aborted.

In a peer-peer strategy the transaction execution is distributed among mobile hosts. In these situations, every mobile host must possess the capability of managing transaction execution and integrated successfully with the fixed hosts. Further in mobile peer-peer systems, each host has to carry its own local database which is a challenge.

5.3 Concurrency control to support update transactions (CCUD)

In mobile commerce, browsing or querying database is not sufficient as some of the applications allow users to update the database. Local validation helps in achieving shorter response time, early detection of conflicts and reduced validation load on the fixed host [7]. However the broadcast errors might lead to ambiguity in making the commit decision. If a mobile host doesn't receive commit information correctly, it couldn't determine whether the local transactions must be committed or not. These broadcast errors can be reduced using reliable transmission protocol like a typical re-transmission and acknowledgement technique.

Two types of broadcasts are used in this strategy viz. Notification broadcast for delivering the validation results of the transactions and Certification broadcast for broadcasting the invalidation information. In addition a confirmation protocol is used by mobile clients to receive the reliable results over a broadcast channel.

The throughput of this strategy is less as it has to go through one additional level in making a final commit decision. This also increases the number of certification broadcast when many mobile hosts are requesting for the same shared data item as a result the uplink bandwidth increases.

5.4 Other optimistic strategies

In a mobile environment if the transaction validation is done on the server, it may lead to delayed response causing overhead at the server [8]. 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 [14]. In [10,19], the conventional optimistic concurrency control algorithm in enhanced with an early termination mechanism on conflicting transactions. But because of early termination a transaction need to be initiated again and again.

Optimistic concurrency control protocols (OCC) [13, 15, 17] are non-blocking and deadlock-free, which make them efficient to use in mobile computing and have been adopted in the Disconnected Operation [11] and Kangaroo Transaction model [16]. But, 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 [18, 19]. In these approaches if the conflict rate increases, more and more transactions get aborted. In [14], author proposes A Timestamp-Based Optimistic Concurrency Control for Handling Mobile Transactions. However it [9,12,14] needs broadcasting of messages to send the invalidation reports which might unnecessary flood the network thereby reducing the transaction throughput. The approach followed in [14] uses broadcasting which is not suitable for some of the applications because of the rise in abort rates and unnecessary flooding of invalidation reports. For example in [1], the author proposes a scheme, to provide non-blocking protocol with restrained communication. But it faces the problem of time lag between the local and global commit. In [10] concept of non-conflicting transactions is introduced such that if a conflicting transaction is detected, it may be aborted.

5.5 Issues & challenges

Optimistic commit protocol can be implemented for transaction processing in mobile environments. However the traditional optimistic protocol doesn't give good performance if it is implemented in mobile environments

The invalidate report is sent to all the participants involved in some transaction execution. (The invalidation report may be sent only to the participants using the same shared data item and the data item is invalidated). Further in the traditional approach, once the data item becomes invalidated, a new request is to be initiated for execution of a transaction. In mobile environments this increases the uplink bandwidth.

The qualitative analysis of the optimistic strategies for guaranteeing concurrency control in mobile environments is presented in table 3.

Table 3. Qualitative analysis of Optimistic Strategies for Concurrency Control

Optimistic Strategy	Trans Type	Data Validity	Conflict resolution	Uplink Bandwidth
CCM	Nested	Till the timer expires (can be utmost for limited no of concurrent request)	Through broadcasting of new values	Low, but less transaction throughput
SODA	Nested	Till the time of writing the write set onto fixed host	Through broadcasting done by mobile hosts (peers)	High, if the transaction fails in third phase, it has to be requested again.
CCUD	Flat	Data Item becomes valid when certification broadcast is invoked	Through broadcasting after local validation	Increases exponentially when access to shared data items increases because of the additional confirmation protocol
Time stamp based OCC	Flat	Till the invalidation report is received	Through broadcasting of invalidation reports	Since data items are read only on demand uplink bandwidth is sufficiently low
Improving CC in Mobile Databases	Flat	Till the invalidation reports are received	Conflicting transactions are terminated before reaching the validation phase	High as the transaction has to be started again once it receives invalidation report

6. CONCLUSION

Concurrency control forms a basic building block for transaction management in any database environments. However as disconnections are treated as normal situations in mobile environments, there is a need for relaxation of ACID properties in mobile environments. In this paper several concurrency control strategies to achieve the Isolation is discussed. In pessimistic strategy the transaction may be blocked due to locking. However the time based strategies made an attempt to resolve the starvation issues in presence of disconnections and mobility. To overcome the problem of blocking, optimistic strategies are presented in literature. However there is a need for good conflict resolution strategies which satisfies Serializability property.

7. REFERENCES

[1] Patrical Serran-Alvarado et.Al, “A Survey of Mobile Transactions”, Distributed & Parallel Databases, Kluwer Publishers, 16, pp.193-230, 2004

[2] K.P. Eswarn, J. Gray, R.A. Lorie, and I.L. Triger, “The notions of consistency and predicate locks in a database system,” Communications of the ACM (CACM), vol. 19, no. 11, 1976.

[3] Kam-Yiu Lam, Tei-Wei Kuo et.Al, “Concurrency Control in Mobile Distributed Real-Time Database Systems”, Elseiver Sciences, Information Systems Vol. 25, No.4, pp. 261-286, 2000.

[4] R.J. Abbott and H. Garcia-Molina. “Scheduling real-time transactions: a performance evaluation. ACM Transactions on Database Systems”, Vol. 17, No. 3, pp. 513-560, 1992.

[5] Nitin Prabhu, Vijay Kumar et. Al, “Concurrency Control in Mobile Database System”, Proceedings of 18th IEEE International Conference on Advanced Information Networking and Application (AINA),pp. 83-86, 2004.

[6] Zhaowen Xing, Le Gruenweld, K.K Phang, “SODA: An Algorithm to Guarantee Correctness of Concurrent Transaction Execution in Mobile P2P Databases”, 19th IEEE International Conference on Database and Expert Systems Applications” pp.337-341, 2008.

[7] Zakil Koo, Songchum Moon, “Effects of broadcast errors on concurrency control in wireless broadcasting environments”, pp. 13-21, 2002.

[8] Victor C.S., Kwok wa Lam and Son, S.H., “Concurrency Control Using Time-stamp Ordering in Broadcast Environments”, the Computer Journal,Vol.45, No.4 PP.410-422, 2002

[9] 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.

[10] Minsoo Lee, Sumi Helal, “HiCoMo: High Commit Mobile Transactions”, Distributed and Parallel Databases, 11, 73-92, 2002, Kluwer Academic Publishers

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

[12] Khalil M. Ahmed, Mohammed A Ismail, Navava M. El. Makky, Khaled M. Nagi, “A New Transaction Management Scheme for MobileComputing Environments”, 2003.

[13] H. T. Kung and J. T. Robinson, "On Optimistic Methods for Concurrency Control", ACM TODS, Vol.6, No. 2, June 1981.

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

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

[16] K. Eswaran, J. Gray, R. Lorie, I. Traiger, “The Notion of Consistency and Predicate locks in a database system”, Communication of the ACM, 19 (11): 624-633, 1976.

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

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

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

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

[21] P. Krishna Reddy, Masaru Kitsuregawa, “Speculative Lock Management to Increase Concurrency in Mobile Environments”, MDA’99, LNCS 1748, pp, 82-96, 1996.

[22] Mohammed Khaja Nizamuddin, Dr. Syed Abdul Sattar, “An Improved, Prioritized ConcurrencyControl Scheme with Performance Gain in Mobile Environments”, pp. 34-40, Vol.1, No.1, ARPN Journal of Systems & Software, 2011.

[23] Mohammed Khaja Nizamuddin, Dr. Syed Abdul Sattar, “Data Count Driven Concurrency Control Scheme With Performance Gain in Mobile Environments”, pp. 106-112, JETCIS, Vol,2. No. 2, 2011.

Authors

Dr. Salman Abdul Moiz is a Research Scientist at Centre for Development of Advanced Computing Bangalore. He received his B.Sc (Electronics) from Osmania University, MCA from Osmania University, M.Tech (cse) from Osmania University, M.Phil (CS) from Madurai Kamaraj University and Ph.D (CSE) from Osmania University. His research interest includes Mobile databases, Domain specific component design & Disaster recovery.

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 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

Jitendra Kumar is working as Senior Staff Scientist at Centre for Development of Advanced Computing, Bangalore. He holds Bachelors degree in Computer Science & Engineering from Magadh University & PGDIM IGNOU, Delhi. His area of interests includes Mobile computing & Applications, SOA, Web Security, Messaging middleware, Enterprise Application Development and Distributed Computing.

Palani Lavanya is working as Staff Scientist at Centre for Development of Advanced Computing, Bangalore. She received B.E (CSE) from Madras University. Her research interests are in the areas of Mobile Computing, Network Security & Embedded Networking.

Deepak Chandra Joshi is working as Staff Scientist at Centre for Development of Advanced Computing, Bangalore. He received B.Sc from Kurukshetra University & MCA from IGNOU. His research & development interest includes Mobile computing Applications, Web technologies, Open source software and SOA applications.

G. Venkataswamy is working as Staff Scientist at Centre for Development of Advanced Computing, Bangalore. He received his MCA from Osmania University. His research interest includes Mobile computing, Web engineering, Distributed Computing, Computer Networks, Databases and SOA.