Volume 1 No. 1, APRIL 2011

ARPN Journal of Systems and Software

http://www.scientific-journals.org

An Improved, Prioritized Concurrency Control Scheme with Performance Gain in Mobile Environments

Mohammed Khaja Nizamuddin[1], Dr. Syed Abdul Sattar[2]

[1]Associate Professor, Department of CSE, Deccan College of Engineering and Technology, Hyderabad, India.

[2]Dean Academics, Royal Institute of Technology and Science, Chevella, India.

mknizams@yahoo.com , syedabdulsattar1965@gmail.com

ABSTRACT

In a mobile computing environment, clients can access shared data irrespective of their physical location and can be updated by each client independently. This leads to inconsistency of the data. Several concurrency control techniques are proposed in literature to prevent data inconsistency. In this paper we first analyze the existing scheme of concurrency control without locking and justify its performance limitations. A new priority based scheme is proposed in which priority is given to the older transaction whenever conflict arises, there by decreasing transaction re-executions and current load of the database server. Experimental results show performance benefits such as increase in commit rate and decrease in re-execution of the transactions.

Keywords- *Absolute Validity Interval, Priority Value, Semaphore, Mobile Host, Fixed Host*

1. INTRODUCTION

A Mobile Database System (MDS) is a distributed multidatabase client/server system based on Personal Communication Service (PCS) or Global System for Mobile Communication (GSM). There are some differences in GSM and PCS architecture however they do not affect MDS. An MDS provides full database and mobile communication functionalities. A mobile user can initiate transactions from anywhere and at anytime. MDS guarantees their consistency preserving execution.

Transaction management has one important feature called concurrency control to ensure consistency of the database. In mobile computing environment, data management becomes a challenging issue due its limitations i.e. variable bandwidth, frequent disconnections, limited resources on mobile host etc [7]. Several valuable techniques are proposed in literature to provide concurrency control in mobile environments; however most of them are based on locking, time stamp and optimistic concurrency control.

In [10] a new concurrency control technique is proposed based on Absolute validity interval (AVI) value [3]. This scheme has a problem of giving equal priority to all transactions, which leads to more transaction aborts. In [5] we justified that non-priority based transaction executions may result in needless abortion of transactions. Several performance limitations of the scheme were also mentioned.

In this paper we elaborate our simulation results and discuss the performance of the proposed scheme [5] with the help of graphs for specific performance parameters. We also justify performance gain of the proposed scheme for achieving concurrency control in mobile environments by eliminating limitations of the existing scheme, increasing commit rate and decrease in re-execution rate of the transactions.

The rest of the paper is organized as follows. Section 2 reviews existing concurrency control schemes. Section 3 describes mobile database environment. In Section 4 we discuss performance limitations of the existing lockless scheme. Section 5 explores proposed concurrency control scheme. Section 6 specifies performance metrics. Section 7 concludes the paper.

2. RELATED WORK

In mobile database environment, providing consistency of the data items is a challenging issue in case of concurrent access. Various valuable attempts are made in providing solutions for data management in mobile environments.

The conventional two phase locking protocol is not suitable, as it requires clients to communicate continuously with the server to obtain locks and detect the conflicts [8]. An optimistic concurrency control technique has the problem of delayed response [12]. In [13] timeout based Commit Protocol has the problem of time lag between local and global commit and more rollbacks of the transactions due to starvation. In [6] the proposed Mobile 2PC protocol preserves the 2PC principle and minimizes the impact of unreliable wireless communication.

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 [2]. In [1, 4], the conventional optimistic concurrency control algorithm is enhanced with an early termination mechanism on conflicting transactions. In [9] dynamic timer management is used for achieving concurrency

control. This suffers from the problem of frequent rollbacks due to regular expiry of the timer and wastage of computation. In [11] preemptive dynamic timer adjustment strategy is proposed to enhance throughput of the system. This scheme is based on assumption of execution time and is not scalable.

3. MOBILE DATABASE ENVIRONMENT

Mobile database environment (fig.1) consists of Mobile Host (MH), Fixed Host (FH) and Base station. The communication of the MH with the FH is supported by Base station. FH is connected with a wired network and has a Database server. MH may have DBMS module to perform database operations.

In a mobile environment a MH can process its workload in a continuously connected mode or in disconnected mode or in an intermittent connected mode.

In *connectivity mode* a MH is continuously connected to the database server. It has the option of caching required data for improving performance or can request data from the server any time during transaction processing. In d*isconnected mode* a MH voluntarily disconnects from the server after refreshing the cache and continues to process workload locally. At a fixed time it connects and sends its entire cache to the server. The server installs the contents of the cache such a way that global consistency is maintained. *Intermittent connected mode* is similar to the disconnected mode, but here the MH can be disconnected any time by the system or voluntarily by the user.

The disconnection by system may be due to lack of channel, low battery, security, etc. The user may disconnect the MH to save power or to process data locally, or no communication with the server is required for some time. Hence disconnections are treated as normal events and not as failures [8]. Unlike disconnected mode, intermittent mode does not have any fixed time for connecting and disconnecting a MH.

In our scheme we assume that transactions copy all data items required from FH to local physical memory of the MH and execute locally. The updates are recorded on the FH in Write through manner i.e., whenever the data item is updated on MH at the same time the update is sent to FH for storing the new value of the data item. For maintaining cache coherency, invalidation report technique is used. In this whenever a data item is updated on the FH, this is informed to the clients, which has a copy of this data item by preparing a report. This report called invalidation report consists of data item-id of all those data items, which are updated from the time last invalidation report is received.

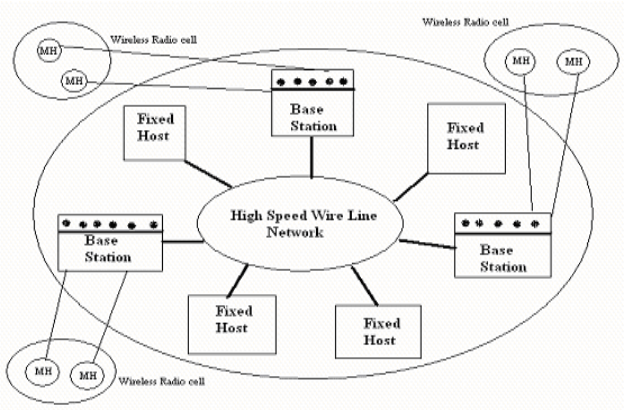

Figure 1 Mobile Database Environment [8]

4. PERFORMANCE LIMITATIONS OF EXISTING SCHEME

- All transactions are given equal priority when trying to access the data item.
- According to principle of serializability, to ensure consistency of concurrently executing transactions, the transaction, which accessed the data item, first should also use this first among others. This principle is not ensured, even though consistency is maintained by simply aborting one of the conflicting transactions.
- When aborted, the work done by the aborted transaction is wasted. Re-execution of the aborted transaction increases load on the fixed host and computation cost also increases.
- Since priority is not maintained, the transaction, which has done most of the work and about to complete its execution, may get aborted.

5. PROPOSED CONCURRENCY CONTROL SCHEME

5.1 Proposed Scheme Features

1. We incorporate a parameter called PRIORITY VALUE (PV), which counts a transaction's access of a data item for write operation.
2. Whenever there is a conflict in data item access, transaction that has higher value of PV (i.e. higher priority) for that data item will be given access. The other is kept in waiting queue.
3. When a transaction updates a data item, transaction's PV of that data item is set to zero.
4. Read and Write operations of a transaction are clearly separated to ensure serializability.

The data stored on the fixed host has the same format [10] to control concurrent access.

Data-Id	Semaphore	TLU	AVI	DATA

Figure 2 Data Item Format on Fixed Host [10]

1. Data-ID denotes a unique-Id of the data item.
2. Semaphore denotes a binary variable, which has either 0 or 1 value.
3. TLU denotes the latest time at which the data item was updated.
4. AVI denotes the absolute validity interval, i.e. the time period for which the data item value is valid [3][10].
5. Data denotes the current value of the data item.

5.2 Proposed Client (MH) Algorithm

1. MH Connects to the fixed host. Checks the availability of all the data items required by comparing its semaphore value to zero.
2. MH then encounters any one of the two cases.
   Case (A): If semaphore=0 is true, MH copies the data item format.
   Case (B): If semaphore=0 is false, then wait in the queue till the data item is available.
3. Repeat step (2) for all data items.
4. Start execution of the transaction locally on the MH.
   Case (A): for read operation of the data item directly execute step (5).
   Case (B): for write operation of the data item check the following equation:

   Current access time> timestamp + AVI ----- (1)
   (Timestamp is the time at which the data item is read by the mobile host)

   i. If equation (1) is true it means that the present value of the data item in private memory is invalid. Since the validity period given by the fixed host as AVI value of the data item is exceeded. Mobile host assumes that the data item might have updated on fixed host. Now the data item cannot be used. The step (1) is re-executed.

   ii. If equation (1) is false it means that the present value of the data item is still valid.
5. Search the Data-Id of the data item in the latest invalidation report.
   Case (A): If Data-Id exist, it means that the value of the data item is updated on the server and the current value is invalid. Execute from step (1).
   Case (B): If Data-Id does not exist, it means that the value of the data item is still valid.
6. Write data item locally and to maintain consistency, update the value of the data item on the server by write through policy. Write through policy is, whenever the data item is updated in the local memory at the same time it should also be updated on the server.
7. If all data items are updated on the Fixed Host then transaction on Mobile Host commits.

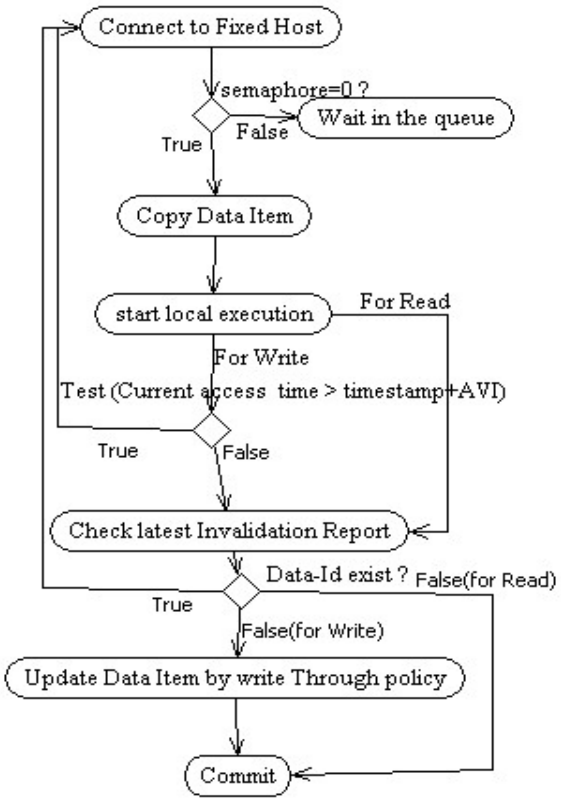

Figure 3 Activity Diagram of MH

5.3 Proposed Server (FH) Algorithm

1. Create the data item format for each data item by initializing semaphore, Data-Id and TLU. Set AVI depending on nature of data item.
2. Create priority table and set Priority Value to zero (PV=0) for each data item.
3. Start timer.
4. Wait for data item request from MH.
   If request arrives check semaphore value of the data item and PV.
   Case (A): If Semaphore=1, add this transaction in waiting queue.
   Case (B): If Semaphore=0, allow the MH to read the data item. If the MH wants to perform a write operation then check Priority table for PV of this transaction.
   i) If PV is highest among other requests then set semaphore=1, Increment PV.

   ii) If PV is less among other requests then Increment PV and add this transaction in waiting queue.

5. Wait for data item update or expiry of AVI.
6. **Case (A):** If data item is not updated before the expiry of the AVI then fixed host set the semaphore value of that data item to 0. New AVI is then dynamically adjusted.

Case (B): If data item is updated before the expiry of the AVI, generate invalidation report by including the data-id and new TLU of the data item and send only to those mobile hosts which have a copy of this data item. New AVI is then dynamically adjusted. Set PV of the transaction and semaphore value for that data item to 0.

7. Fixed host now evokes next transaction from waiting list.

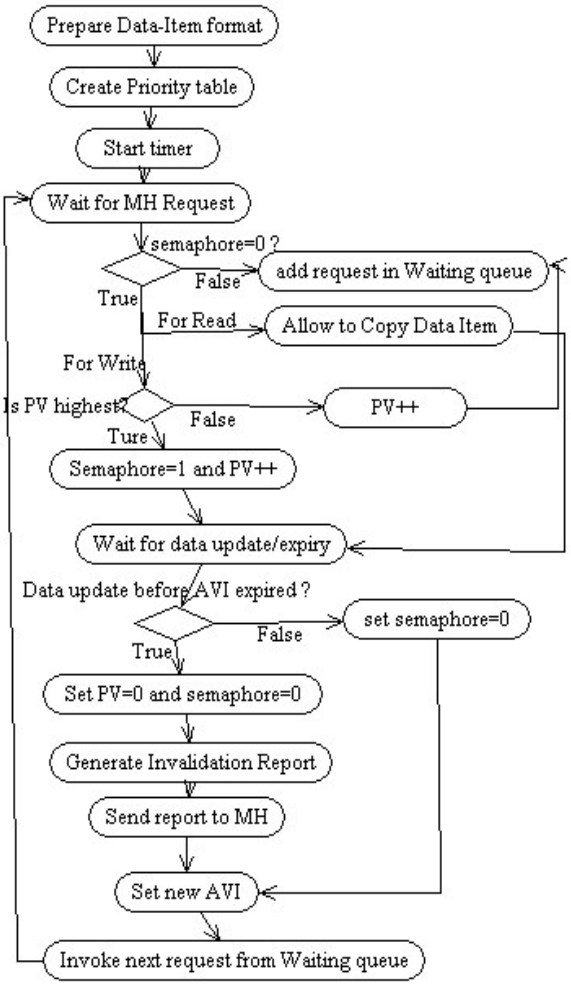

Figure 4 Activity Diagram of FH Server

6. PERFORMANCE METRICS

6.1 Simulation Detail

Same concurrent execution scenario of the transactions as presented in [10] is taken to show performance enhancement. The table.1 depicts execution steps of the transactions. X, Y, Z are the data items. Each Mobile Host (MH) requires two data items. Copy denotes the operation of copying the data item from Fixed Host (FH) to MH. Read, Write, commit are the operations of normal transaction execution.

Table 2. represents the given Mobile Host transactions execution schedule by the existing scheme [10]. The Current Access Time depicts the time of the each logical operation of transaction execution. RW (A) represents copying the data item into the private memory and performing the write operation on A. R (A) represents copying the data item into the private memory for performing read only operation. WAIT (A) specifies that the mobile host is waiting to acquire the data item A. WRITE (A) denotes write through policy for updating the data item A. A (T) denotes the time T at which the data item A is copied by the Mobile Host.

Table 1 Execution Steps of Transactions

MH1	MH2	MH3
Copy (X)	Copy (Y)	Copy (Z)
Copy (Y)	Copy (Z)	Copy (X)
Write (X)	Read (Y)	Write (Z)
Write (Y)	Write (Z)	Write (X)
Commit	Commit	Commit

MH=Mobile Host

Table 2 Transaction Execution on MH [10]

CT	MH1	MH2	MH3	AVI			Timestamp		
				X	Y	Z	MH1	MH2	MH3
11	RW (X)	R (Y)	RW (Z)	5	6	4	X (11)	Y (11)	Z (11)
12	RW (Y)	WAIT (Z)	WAIT (X)	5	6	4	Y (12)	--	--
13	WRITE (X)	---	---	3	6	4	--	--	--
14	WRITE (Y)	Inv.Report	RW (X)	3	5	4	--	--	X (14)
15	COMMIT	RW (Z)	ABORT	2	5	7	--	Z (15)	--
16	---	R (Y)	---	-	5	7	--	Y (16)	--
17	---	WRITE (Z)	---	-	-	5	--	--	--
18	---	COMMIT	---	-	-	-	--	--	--

CT=Current Access Time, AVI=Absolute Validity Interval

In the above schedule MH3 aborts due to equal priority of all transactions for a data item. Since by the time it updates 'Z' after waiting for it, transaction MH2 acquires it violating Serializability. Hence MH3 aborts execution.

Table 3 represents Mobile Host transactions execution schedule with the proposed scheme. MH3 now commits due to more priority value for the data item 'Z'. MH3 has acquired data item 'Z' first, it should use this first to maintain serializability. With a cost of 2 time units the system is able to give full commit rate, though some waiting time of the transaction may increase. The results of the scheme over increased number of transactions given more commit rate.

Table 3 Transaction Execution on MH [5]

CT	MH1	MH2	MH3	AVI			Timestamp		
				X	Y	Z	MH1	MH2	MH3
11	RW (X)	R (Y)	RW (Z)	5	6	4	X (11)	Y (11)	Z (11)
12	RW (Y)	WAIT (Z)	WAIT (X)	5	6	4	Y (12)	--	--
13	WRITE (X)	---	---	3	6	4	--	--	--
14	WRITE (Y)	Inv.Report	RW (X)	3	5	4	--	--	X (14)
15	COMMIT	WAIT (Z)	RW (Z)	3	5	7	--	--	Z (15)
16	---	---	WRITE (Z)	3	5	6	--	--	--
17	---	RW (Z)	RW (X)	5	5	6	--	Z (17)	X (17)
18	---	R (Y)	WRITE (X)	4	5	6	--	Y (18)	--
19	---	WRITE (Z)	COMMIT	4	7	5	--	--	--
20	---	COMMIT	---	4	7	5	--	--	--

The server updates the values of the data item format only when some transaction uses it. Table 4 represents data item format values stored on the server while concurrent execution of the transactions.

Table 4 FH Data Item Status While Execution [5]

CT	Semaphore Value			AVI			*TLU*		
	X	Y	Z	X	Y	Z	X	Y	Z
10	0	0	0	0	0	0	0	0	0
11	1	0	1	5	6	4	0	0	0
12	1	1	1	5	6	4	0	0	0
13	0	1	1	3	6	4	13	0	0
14	1	0	0	3	5	4(exp)	13	14	0
15	1	0	1	3	5	7	13	14	0
16	0	0	0	3(exp)	5	6	13	14	16
17	1	0	1	5	5	6	13	14	17
18	0	0	1	4	5(exp)	6	18	14	17
19	0	0	0	4	7	5	18	14	19
20	0	0	0	4	7	5	18	14	19

TLU=Time of Last Update

Table 5 represents priority table created and maintained by FH to provide access to data items by each transaction.

Table 5. FH Priority Table While Execution [5]

CT	PV for MH1			*PV for MH2*			*PV for MH3*		
	X	Y	Z	X	Y	Z	X	Y	Z
10	0	0	0	0	0	0	0	0	0
11	1	0	0	0	0	0	0	0	1
12	1	1	0	0	0	0	0	0	1
13	0	1	0	0	0	0	0	0	1
14	0	0	0	0	0	0	1	0	1
15	-	-	-	0	0	0	1	0	2
16	-	-	-	0	0	0	1	0	0
17	-	-	-	0	0	1	2	0	0
18	-	-	-	0	0	1	0	0	0
19	-	-	-	0	0	0	-	-	-
20	-	-	-	-	-	-	-	-	-

PV=Priority Value

6.2 Performance Analysis

The simulation is done using Java network programs and the performance is studied by NS2. For simplicity all transactions work set is taken as homogeneous i.e. all transactions have equal number of operational steps and only half of the data items are shared data items.

The performance graphs of existing scheme (AVI based) and proposed scheme (PAVI based) are drawn using the data obtained in simulation. Since Commit rate and Re-Execution rate of transactions are two essential parameters of performance, the graph is drawn and studied only for the two parameters. The load of the system is increased incrementally.

i) Total Transactions Vs Transaction Commit Rate

The graph shows implementation results of both algorithms for commit rate. Commit Rate of the transactions is the number of transactions completed execution from the total number of transactions in execution. In other words for every sample of experiment how many transactions successfully committed execution from the total number of transactions taken for concurrent execution.

The graph clearly shows increase in commit rate for the proposed scheme. Since priority is given for the transaction, which has first requested the data item, and to ensure serializability it is given precedence over the other requests. This has increased the chances of commit of the older transaction. Hence more number of transactions gets executed without abort due to violation of serializability.

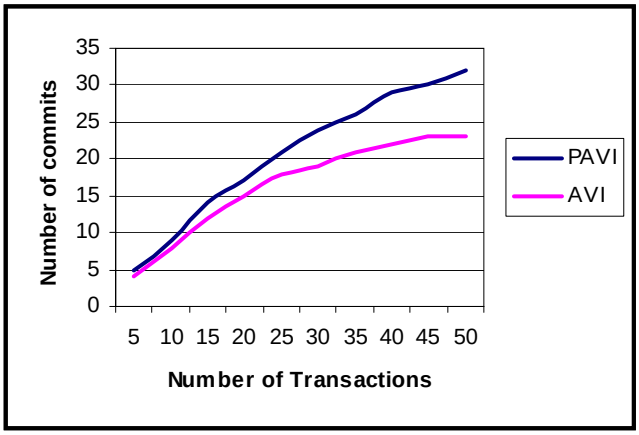

Figure 5. Performance Graph of Commit Rate

The graph also shows that as the load of the system increases, the proposed scheme gets very low effect. Where as the AVI scheme gets affected badly when the load of the system increases and very less number of transactions commit execution. It is also important to note that the proposed scheme gives more than 60% commit rate, where as the AVI scheme gives less than 50% commit rate as the load of the system increases.

ii) Total Transactions Vs Transaction Re-Execution Rate

The graph shows implementation results of both algorithms for Re-Execution rate. Re-Execution rate of the transactions is the number of transactions re-execution from the total number of transactions in execution. In other words, for every sample of experiment, how many transactions re-executed due to violation of serializability before final commit from the total number of transactions taken for concurrent execution.

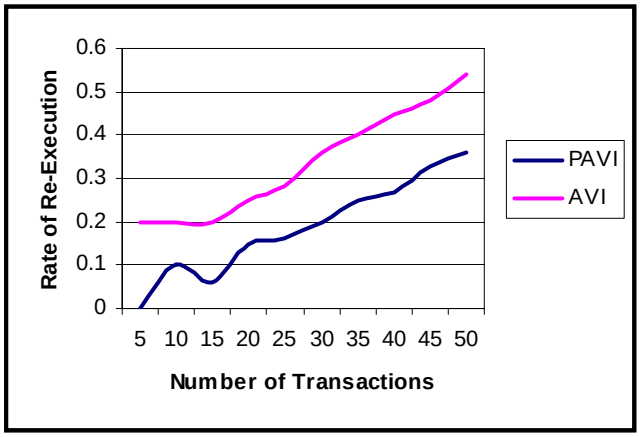

Figure 5 Performance Graph of Re-Execution Rate

It is evident from the graph that the proposed scheme effectively reduces the re-execution rate of the mobile transactions. This is because the proposed scheme gives more priority to the transaction, which requested the data item first in the execution while resolving conflict of data item access. This ensures implicit serialization order of the concurrently executing transactions, which is essential to maintain consistency of the database. Whereas in AVI scheme serializability is ensured by re-executing the transaction which has violated serialization order. Since in AVI scheme all transactions are treated equally. Therefore more number of transactions gets re-executed compared to the proposed scheme.

6.3 Advantages of Proposed Scheme

1. Transactions data item access priority is maintained to ensure serializability without aborting the transactions.
2. The cost of waiting time of the transaction to execute is less than the cost of re-execution of the transaction. Hence, transaction can wait little more to acquire a data item than to access and get aborted.
3. The transaction, which has done more work, is given higher priority, as it will finish early if given more privilege.

4. The overall through put of the system increases by sacrificing a small amount of waiting time and bandwidth is conserved.

7. CONCLUSION

In this paper we have shown the limitations of the existing concurrency control scheme. We also justified that assigning priority to transactions for a data item ensures serializability without aborting the transaction. The proposed concurrency control scheme improves the drawbacks of the existing scheme by prioritizing the transactions based on the work done. Due to very few transaction re-executions current load of Fixed Host server is reduced. It also increases the overall commit rate of the system and decreases the rate of re-executions.

ACKNOWLEDGMENT

The work reported in this paper is carried out as a Research scholar of Rayalaseema University, Kurnool, India. The author is very much thankful to the University for Provision of the environment in which research was conducted.

REFERENCES

[1] Anand Yendluri, Wen-Chi Hou, Chih-Fang Wang,"Improving Concurrency Control in Mobile Databases",DASFAA 2004, LNCS 2973, pp. 642-655, 2004.

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

[3] Joe Chun-Hung Yuen et Al. "An Adaptive AVI-based Cache Invalidation Scheme for Mobile Computing Systems", DEXA 2000, pp. 155-178, IEEE Computer Society, USA.

[4] Minsoo Lee, Sumi Helal, "HiCoMo: High Commit Mobile Transactions", Distributed and Parallel Databases, Vol.11, pp.73-92, 2002, Kluwer Academic Publishers.

[5] Mohammed Khaja Nizamuddin, Dr. Syed Abdul Sattar, "Algorithm for Priority Based Concurrency Control Without Locking in Mobile Environments" 3rd IEEE International Conference on Electronics Computer Technology. ICECT 2011, in press.

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

[7] Patricia Serrano-Alvarado, Claudia Roncancio, Michel Adiba, "A Survey of Mobile Transactions", Distributed and Parallel Databases, Vol.16, 193-230, 2004, Kluwer Academic Publishers.

[8] Salman Abdul Moiz, Dr. Lakshmi Rajamani, "An Algorithmic approach for achieving Concurrency in Mobile Environment", INDIACom, 209-211, 2007.

[9] Salman Abdul Moiz, Dr. Lakshmi Rajamani, "Single Lock Manager Approach for Achieving Concurrency in Mobile Environments", Proceedings of 14th IEEE International Conference on High Performance Computing, 2007. Springer LNCS-4873, 650-660, 2007.

[10] Salman Abdul Moiz, Mohammed Khaja Nizamuddin, "Concurrency Control Without Locking in Mobile Environments", Proceedings of IEEE 1st International Conference on Emerging Trends in Engineering & Technology, pp1336-1339, 2008.

[11] Salman Abdul Moiz, Dr. Lakshmi Rajamani,,"Concurrency Control Strategy to Reduce Frequent Rollbacks in Mobile Environments," cse, vol.2, pp.709-714, 2009 International Conference on Computational Science and Engg, 2009.

[12] Victor C.S., Kwok, W.L and Son, S.H., "Concurrency Control Using Timestamp Ordering in Broadcast Environments", The Computer Journal, Vol.45 No.4 PP.410-422, 2002.

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