CWHP algorithm for Scheduling Real-Time Transactions

Fadia A. Elbagir#1, Ahmed Khalid*2, Khalid Khanfar#3
#1 PhD. Program in Computer Science, Sudan University of Science and Technology, Sudan
#2 Department of Computer Science, Community College, Najran University, Najran, KSA
#3 Full professor Head of Information Security Department at Naif Arab University for Security, Saudi Arabia

Abstracts

Scheduling is an important issue in the designof real time database systems; Transactions in real time systems must be scheduled in such a way that they can be completed before their deadlines, the scheduler assigns a priority to each transaction based on its deadline, we are particularly interested in conflicts that can lead to priority inversions.Priority inversion problem may occur due to the sharing of resources among transactions,which can cause unbound delay to high priority transaction; this delaying may result in the higher priority transactions missing their deadline.In this paper we proposed a new scheduling algorithm Conditional Waiting High Priority (CWHP), we used simulation model tocompare the performance results of our algorithm with other existing algorithms using of the most popular priority assignment schemes Earliest Deadline First (EDF) policy, focusing in "firm deadline" real time applicationsand timing information about firm transactions where transactions that miss their deadlines are discarded and the objective of the real-time database system is to maximize the number of transactions that satisfied deadlines.

Keywords**:** *Real Time, Scheduling, Firm Deadline, Priority inversion, transactions, Earliest Deadline First*

I. Introduction

A real-time database system (RTDBS) is a transaction processing system that is design to handle workloads where transactions have completion deadlines [10,11,35].For scheduling transactions to satisfy time constraints and data consistency requirements, the efficient scheduling algorithm and concurrency control protocols are requiredto induce a serialization order among conflicting transactions [26]. For a concurrency control protocol to accommodate timing constraints of transactions, the serialization order it produces should reflect the priority of transactions. [2,5]
The execution of concurrent transactions is scheduled based on their assigned priority. [2, 20].The conflict is appears when two transactions request the same data item and at least one of them is an exclusivelock,Under two phase locking protocol a transaction must obtain a lock before accessing a data object and release the lock when it commits or aborts[4],Ideally a high priority transaction should never be blocked by any lower priority transaction, the transaction which requesting a lock on data item be placed in a wait queue if its lock mode is found to be incompatible with that of the lock-holding transaction.[4,6,20]
In real-time database systems, blockingmay cause priority inversion. Priority inversion is said to occur when a high priority transaction is blocked by lower priority transactions [2, 4, 20]. The alternative is to abort or restart lower priority transactions when priority inversion occurs. This wastes the work done and very costly in terms of wasted resources, and a large number of restarts will increase the workload of the system and may cause other transactions to miss their deadlines and in turn also has a negative effect on time-critical scheduling.[10,13,32]
In this paper we investigate the priority inversion problem in a particular real-time environment and proposed new algorithm **"Conditional Waiting High Priority (CWHP)"** for scheduling transactions,according to the priority assignmentpolicy"**Earliest Deadline First"**.

The rest of this paper is organized as follows: Section II introduces related works. Section III Describe Scheduling and Concurrency Control. In section IV the Implementation ofAssigning Priorities.Section VDescribes the proposed methods and comparison with the previous methods, section VI described the simulation model & result.

II. Related work

Many researchers were discussing and published papers for the issue in real time database systems and scheduling.

In [3] Authors focused on scheduling transactions with deadlines and the effect of real time constraints on concurrency control. they discussing a new group of algorithms for scheduling real time transactions which produce serializable schedules, in result of evaluated the scheduling through detailed simulation experiments they found that Earliest Deadline (ED) is the best policy

overall priority assignment policy, and There is one case where Serial execution (SE) may be superior to Conditional Restart (CR). This occurs when there is a high cost for restarting transactions. However, SE does not become a better method until the cost of restarting is more than half of the computation time of average transactions.

In [4] Robert Abbott addressed modeling of real time constraint and scheduling problems,they handled simulation experiment to study performance and behavior of variety of scheduling algorithms focused on the effect of cognizant algorithm on the concurrency of run times estimates and in the result of their work they conclude that their simulation studies will enable to develop better heuristics for scheduling real time transactions.

In [7] Ben Kao1, and Hector Garcia-Molina, discussed the various issues concerning the design and implementation of real-time databases and transaction processing,they discussing the role of application semantics and showed how they can be used to improve RTDBSs performance. They gave a brief note for that appropriate deadline assignment to transactionsthat is very critical to the success of many real-time database protocols.

In [18]Authors address the problem of establishing a priority ordering among transaction characterized by both value and deadline that results in maximizing the realized value. They study the relationswhich established between these values and deadlines in constructing the priority ordering. they evaluate the performance of several priority mapping by simulationmodel,they conclude that a" bucket" priority mechanism allows the relative importance of values and deadlines to be controlled introduced and studies.In conclusionof the study is that there earlier results generally carry over the value-based RTDBS domain for all the priority mappings that we have considered.

In [28] the authors found that in the environment where the physical resource is limited, a concurrency control algorithm that tends to conserve physical resources by blocking transactions that might otherwise have to be restarted is a better choice than a restart-oriented algorithm, and they found the optimistic algorithm to perform the best of the three algorithms tested under these conditions. They reconfirmed there result with other studies, However, the overall conclusions about which algorithm performed the best relative to the other algorithms were not altered significantly by this assumption

Sang H. Son& others in [32] present an intelligent dynamic scheduling algorithm for transactions in real-time database systems. The scheduling algorithm uses timing information about transactions and the database to enhance the system's ability to meet transaction deadlines. The scheduling algorithm is implemented in a simulated puke detection system, and its performance is demonstrated by a series of experiments. The proposed algorithm has been implemented and evaluated using a pulse detection system as a real-life, real-time database application. The experimental results show that scheduling algorithms for real-time database systems can be made more effective by making use of extra information about transactions and the database,the dynamic scheduling algorithm presented in this paper shows promising characteristics that are important to the problem of real-time transaction scheduling.

III. Scheduling and Concurrency Control

The main goal of scheduling in RTDBS is to meet timing constraints and to enforce data consistency. Real-time transactions scheduling methods can be extended for real-time transaction scheduling, while the concurrency control protocols are needed for operation scheduling to maintain data consistency [31,32]

The concurrency control of transactions in a real-time database must satisfy timing constraints of individual transactions and the consistency constraints of the database. Various scheduling algorithms have been developed to schedule real timetransactions to meet their timing constraints,given the arrival time deadline, execution time and criticality of each task. [14, 26, 26, 37]

In case of concurrently executed transactions we need a concurrency control mechanism to order the updates to the database so that the final schedule is a serializable,an Efficient resource scheduling algorithms and concurrency control protocols are required to schedule RTDB transactions to maximize the number of satisfied deadlines [8,23,25,30]

In general many researchers note that the scheduling transactions according to the Earliest DeadlineFirst (EDF)policy in which priorities are based on transaction deadlines can minimize the number of late transactions especially when the system is highly loaded.[7,16], there for we found most research studies in real-time databases with hard deadlines have taken the earliest-deadline first approach for scheduling priorities , mention that Amongst the proposed resource scheduling algorithms for RTDBS .[2,10,27,15]

In general, the performance of real-time concurrency control system is affected by the method used for assigning the priorities of the transactions [37]

Our method implementation depends on comparing transaction priorities at the time of the conflict, to take appropriate decisions.

IV. Assigning Priorities

As considered by A. P. Buchmann Priority scheduling is a mechanism for including a limited measure of timeliness in blocking concurrency-control mechanisms [1]. Priorities are calculated based on deadlines the transaction considered as a higher priority as it closer the deadline, If the deadline of transactions is passed,howeverexecuted, either are dropped, or executed later [1, 5, 7, 15, 24]

The priority assignment usually takes into account the deadlines of the transactions because the underlying assumption is that the deadline reflects the urgency of completing the transaction. Scheduling policies such as earliest deadline first (EDF) and least slack first (LSF) are examples of policies that account for deadlines [2, 19, 5, 37].

First Come First policy assigns the highest priority to the transaction with the earliest release time. The primary weakness of FCFS is that it does not make use of deadline information. FCFS will discriminate against a newly arrived task with an urgent deadline in favor of an older task which may not have such an urgent deadline. This is not desirable for real-time systems.

Earliest Deadline policy assigns highest priority for the transaction with the earliest deadline. A major weakness of this policy is that it can assign the highest priority to a task that has already missed or is about to miss its deadline.

Least Slack policyhere a transaction's priority depends on the amount of service time that it has received. Rolling back a transaction to its beginning reduces its effective service time to 0 and raises its priority under the Least Slack policy.

The scheduler is invoked whenever a transaction terminates and, for preemptive scheduling, whenever a new transaction arrives. The concurrency control mechanism is invoked to resolve lock conflicts whenever one occurs.

V. The Model

Our objective by proposed algorithm is to minimize the number of transactions that miss their deadlines, our proposed methods is a continuation of previous algorithms that proposed in [3,4]; we assume that transaction executions must be serializable by using a locking protocol. In case of transaction arrives, the concurrency control mechanism is invoked to resolve lock conflicts whenever one occurs.

When transaction enters to the system characterized by its timing constants and its data and computation requirements, the time constraints is a release time anddeadline,computationrequirement is anestimation run time which considered as amount of computation time required by the transaction.

Each arriving transaction has a release time r (Arrival time) , Estemated run time E ,and deadline D which is acount as maximum commit time.[25,35]

- $DT = AT + ST + RTT$

- Where,
- DT = Deadline of Transaction
- AT = Sink Arrival time of Transaction
- RTT = Resource Time of Transaction
- ST = Slack Time

As discussed in [3] there four techniques to resolve conflicts that may lead to a priority inversion. In the following discussion let T_R be a transaction that is requesting a lock on a data object O that is already locked by transaction T_H. Furthermore, the lock modes are conflicting and T_R has a higher priority than the priority of O. Thus the priority of T_R is greater than T_H. In our experiment example for tested the new method and Compared with previous methods, we considered the set of transactions with release time r, deadline d, runtime estimate E and data requirements.

Method1: Wait techniques with earliest deadline first for resovling the conflict
Resolution for Wait policy: Requesting transaction always blocks and waits for the data object to become free. **[3,4,7,36,37]**

IF TR conflict with TH

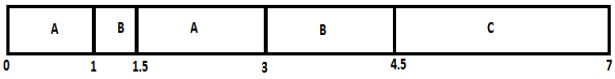
Then TR Blocks

I. TABLE 1

EXAMPLE1

transaction	Arrival time (r)	Execution time (E)	Deadline (d)	updates
A	0	2.5	5	X
B	1	2	4	X
C	2	2.5	8	Y

- Transaction A is the only job in the time 0, it gains the processor and executes until time 1 when transaction B arrives
- T_B during this time it request and gain an exclusive lock on data object X since T_B has an earlier line than T_A
- T_B prempts T_A and begin to to execute
- At time 1.5 T_B attempts to lock data object X which already locked by A
- Under Wait strategy B must wait untIl T_A is finishes and releases the lock on X thus B loses the processor
- T_A resumes execution and completes its remaining 1.5 units of computation at time 3 , when it commits it releases the lock on item X thus T_B is unblocked and resume execution ,it finish its remaining 1.5 units of computation at time 4.5
- T_C arrives and preemptes ,it has late deadline than T_B.
- T_C executes to completion and finishes at time 7
- Under this schedule ,T_B misses its dedline by .5 time units and T_A and T_C both meet there deadline as shown in figure 6
- The over all schedule length is 7



Method2: High Priority techniques with earliest deadline first for resovling the conflict
Resolution for High Prioritypolicy: Comparing transaction priorities at the time of the conflict. If the priority of TR is greater than the priority of object O, and thus greater than every transaction holding a lock on O, then we abort the lock holders thereby freeing the object for TR[3,4,7,36,37].

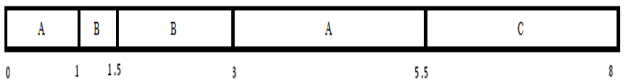
IF For all TH holding a lock on O

P (TR)>P (TH) AND P (TR)>p (T^A_H)

THEN Abort each lock holder

ELSE TR blocks

- A runs in the first time unit during which it acquires a lock on item X
- Transaction B gains the processor at time 1 and causes a conflict at time 1.5 when it requests a lock on item X
- Since B has an earlier deadline than A and thus a higher priority, the conflict is resolved by rolling back A thereby freeing the lock on X
- Transaction B continues processing and completes at time 3
- Transaction A regains the processor and starting from the beginning, executes for 2.5 units and finishes at time 5.5
- Transaction C gains the processor at time 2, starting the execution after Transaction A release the lock and completes at time 8
- In this schedule, A misses its deadline by 0.5 units and B and C meet their deadlines.
- The overall schedule length is 8



Method3: Conditional Restart techniques with earliest deadline first for resovling the conflict
Resolution for Conditional Restart policy: estimate if TH, the transaction holding the lock, can be finished within the amount of time that TR, the lock requester, can afford to wait[3,4,7,36,37].

IFP (TR)>P (TH) AND P (TR)<p (T^A_H)

THEN IF SR ≥ EH -PH

THEN TR blocks

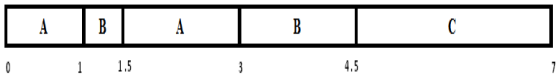
TH inherits the priority of TR

ELSE Abort TH

ELSE TR blocks

- The idea here is to estimate if T_H, the transaction holding the lock, can be finished within the amount of time that T_R, the lock requester, can afford to wait.
- a conflict occurs when B requests a lock on X at time 1.5.
- At this time the algorithm calculates the slack time for B as S = 4 – 2 – 0.5 = 1.5, the remaining run time for A= 2.5- 1= 1.5
- This equals exactly the remaining run time for A.
- Therefore, B waits and A inherits the priority of B.
- Transaction A,executesthen Transaction B is unblocked and resumes execution to finish at time 4.5
- Then C executes to finish at time 7.

- In this schedule B missed it is deadline , A and C transactios meet there deadline
- The overall schedule length is 7



Proposed model: it called **ConditionalWaiting High Priority technieque (CWHP)** to resolve the conflict .

Resolution for Conditional waiting high priority: itComparing transaction priorities at the time of the conflict and if TH, can be finished within the amount of time that TR can afford to wait, if not, it Considered the waiting time for TR as the remaining time for TH.

IFP (TR)>P (TH) AND P (TR)<p (T^{A}_{H})

THEN IF SR ≥ EH -PH

THEN TR blocks

TH inherits the priority of TR

THEN IF (AT_{TR}+RE_{TH} +E_{TR}) ≤ D_{TR}

THEN TR blocks

TH inherits the priority of TR

ELSE Abort TH

The different from new method and Conditional Restart methods as it works as controller which monitors the situation by counting the remains time of TH and replacing this value with the STvalue of TR, in other word all times the waiting time for TR is equaled to the remaining time to complete the process for TH,

- T_B prempts T_A and begin to execute
- At time 1.5 T_B attempts to lock data object X which already locked by T_A
- Under Conditional Waiting High Priority strategy B must wait untill T_A is finishes
- Our proposed algorithm Conditional Waiting High Priority technieque (CWHP) counted the remaining time of execuion time of T_A using these Formula:

RT_A = 2.5-1 =1.5
The slack time of B
$S_B = D_B - E_B$
S_B = 4-2- 0.5 = 1.5
ST_B : is how long transaction execution can be delayed while still making it possible to meet the transaction deadline

$$\mathbf{AT_{TR}+RE_{TH}+E_{TR} \leq D_{TR}}$$

AT_{TR}: Arrival time for transaction that request the lock
RE_{TH}: Remaining time for transaction that holds the lock
E_{TR}: Executiontime for transaction holds the lock
$D_{TR:\ Deadline}$ for transaction that request the lock

The Conditional Waiting High Priority technieque (CWHP) : It work by two ways for silving the conflict :
First : counting the remaining time of execuion time of T_A using these Formula:

$$\mathbf{RET_A = ET_A - \Delta ET_A}$$

RET_A : is the remaining executin time for T_A which holding the lock on data object X
ET_A: execusion estemated time for T_A
ΔET_A : is the amount by serves already T_A recieved
The deadline of transaction B calculated by formula

$$\mathbf{DT_B = AT_B + RET_A + E_B}$$

DT_B: deadline of transaction B
AT_B : Arrival time of transaction B
RET_A: the remaing time for execution transaction A
E_B : execusion time of transaction B
To Calculates how long transaction execution can be delayed while still making it possible to meet the transaction deadline we use ST_B

$$\mathbf{ST_B = D_B - E_B}$$

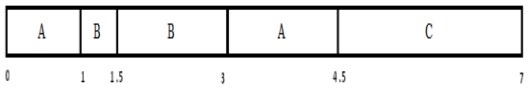
- The CWHP compare the remaining execution time of A RET_A with slack time of S_B
  - **if $RET_A \leq S_B$**
- T_R blocked until transaction T_H release the lock on dada object and T_R resumes execution and untill completes its computation.
-

If we take the previous example for our methods:

- Transaction A is the only job in the time 0, it gains the processor and executes until time 1 when transaction B arrives
- T_B during this time it request and gain an exclusive lock on data object X since T_B has an earlier line than T_A

- The CWHP compare the remaining execution time of A RET_A with slack time of S_B
- if$RET_A \leq S_B$; 1.5 ≤ 1.5
- The deadline of B
- ATB+RTA+ETB ≤ DB
- 1+1.5+ 2= 4.5
- After calculation under the CWHP policy T_B executed and release the lock on dada object X at time 3.
- T_A resumes execution and completes its computation at time 4.5 , when it commits it releases the lock on item X
- T_C enter the system at time 2 and start the execution at time 4.5 and finish at time 8
- The over all schedule length is 7

- By using new propsed CWHP all transactiom meet there deadline as shwon in figure



VI. Simulation Experiments & Result

The simulation testbed constructed using the C# programming language within Microsoft visual Studio Environment

The simulation is the system of queuing network where a numbers of users submit transaction request, any new or re-submitted transaction will enter the scheduling queue and arranged by Early Deadline First priority method. Before transaction perform operation, it must go through the concurrency control (cc) to obtain lock on the objects, if request of lock is denied, the transaction will be placed into wait queue.

Two or more transaction has a data conflict when they were requiring the same data in non- compatible lock model (rr–wr).

our proposed new algorithm compared with existing protocols for Prove it is effectiveness.

The transaction waiting for the lock may terminate or restart depends on the deadline time.

Transaction will be restarted if it still has some value of the system, only if we estimate that the transaction can complete before the deadline.

Each arriving transaction has a release time r (Arrival time) , Estemated run time E ,and deadline D which is acount as maximum commit time.

The performance of a scheduling algorithm depends upon the average utilization of the system. At low utilizations, sufficient time exists for nearly all overrunning jobs to complete in time. As the utilization increases, overrunning jobs become more likely to miss their deadlines.

As we planned that our algorithms should schedule all transactions such that all dead line are met and that transaction with missing deadline minimized, the simulation experiments show that the new algorithm CWHP "Conditional waiting High Priority" reduce the number of deadline missed compared with four other algorithms. We examined the performance of five concurrency control mechanism, each algorithm paired to the priority protocol Early Deadline First, within different workloads such as normal load and heavy load.

For a given set of transactions, order according to which the tasks are to be executed such that various constraints are satisfied.

The simulation experiments using the parameters, and were considered for scheduling as in example1: release time, deadline, and execution time of the transaction. success ratio (commit percentage) was used as measure of performance metrics in our simulation result, it is the percentage of input transactions that the system is able to complete before their deadline.

VII. Conclusion

Transactions in real time systems must be scheduled in such a way that they can be completed before their deadlines, the scheduler assigns a priority to each transaction based on its deadline, in this dissertation we proposed a new scheduling algorithm Conditional Waiting High Priority (CWHP), for enhanced the performance of the system by minimized the number of transaction that is missed their deadline , the algorithm checked transaction priorities at the time of the conflict and it considered the waiting time for transaction that request the lock of resources as same as the remaining time for transaction that hold the data at conflict time. We have used a simulation model for the purpose of comparing the performance results of our algorithm with other existing algorithms that are using the most popular priority assignment schemes **Earliest Deadline First (EDF)** policy.The performance of transactions performed by comparison of transactions commit percentages for each algorithm, our simulation result showed that our proposed algorithm is effective and used it decreased the numbers of transaction that missed their deadline, therefor increase the overall system performance.

A CKNOWLEDGEMENT

*I would like to express my deepest gratitude to myresearch supervisor **Dr. Khalid Khanfer**, for his providing me with an excellent atmosphere I would like to thank **Dr. Ahmed Khalid**, who let me experience and supported my research, Special thanks to Dr. Aiz-Aldin Who provided us with this opportunity and illuminated our path.*

*I am extremely grateful to all **my family** for their love, prayers, caring and sacrifices for educating and preparing me for my future, always there cheering me up and stood by me through the good times and bad.*

References

[1] A. P. Buchmann. D. R. McCarthy. M. Hsu. and U. Dayal, Time-Critical Database Scheduling: A Framework for Integrating Real-Time Scheduling and Concurrency Control, Xerox Advanced information Technoio, Four Cambridge Center, Cambridge MA &!142, Feb ,1989 IEEE

[2] Abbott, R. and Garcia-Molina, H. Scheduling real-time transactions with disk- resident data. Proceedings of the Fifteenth International Conference on l, very Large Database Systems, Amsterdam, 1989.

[3] Abbott, R., And Garcia-M• Lina, H, Scheduling real-time transactions: A performance ACM Transactions on Database Systems, Vol. 17, No. 3, September 1992.

[4] Abbott, R., and Garcia-Molina, H. Scheduling real-time transactions. ACM SIGMOD Rec. (Mar. 1988), 71-81.

[5] ArezouMohammadi and Selim G. Akl, Scheduling Algorithms for Real-Time Systems, Technical Report No. 2005-499, supported by the Natural Sciences and Engineering Research Council of Canada., July 15, 2005

[6] Azer BEST A VROS et. al., Real-Time Database Systems: Issues and Applications, SPRINGER SCIENCE+BUSINESS MEDIA, LLC, Copyright © by Springer Science Business Media New York, 1997

[7] Ben Kao; and Hector Garcia-Molina, An Overview of Real-Time Database System, 1994

[8] C. H. Papdimitriou: The Theory of Database Concurrency Control. Computer Science Press (1986)

[9] E. D. Jensen, C. D. Locke, H. Tokuda: A Time-Driven Scheduling Model for Real- Time Operating Systems. IEEE Real-Time System Symposium. (1985) ,112-122

[10] Haritsa, J et al., Earliest-deadline scheduling for real-time database systems. Proceedings of the Twelfth IEEE Real-Time Systems Symposium, San Antonio, TX, 1991.

[11] Haritsa, J et al., "Data Access Scheduling in Firm Real-Time Database Sys-tems," Journal of Real-Time Systems, vol. 4, Sept. 1992.

[12] HousineChetto& Marline Chetto , Some Result of the Earliest Deadline Scheduling Algorithm,IEEE Transaction on software Engineering vol.15,10 October 1989

[13] Huang, et al., Experimental evaluation of real-time transaction processing. Proceedings of the Tenth IEEE Real-Time System Symposium, Santa Monica, CA, 1989.

[14] ijzgiirUlusoy, A Study of Two Transaction-Processing Architectures for Distributed Real-Time Data Base Systems, J. Systems Software 1995; Systems Software 1995; by Elsevier Science Inc 31:97-1080 1995.

[15] J. Huang & J. Stankovic& et, al., Real-Time Transaction processing: Design, Implementation and Performance Evaluation, COINS Technical Report 90-43. May,1990

[16] J.R. HARITSA et al, Data Access Scheduling in Firm Real-Time, Academic Publishers. Manufactured in The Netherlands. The Journal of Real-Time Systems, 4, 203-241 (1992).

[17] JAUHARI et al., Priority hints: An algorithm for priority based buffer management. TR 911, Computer Sciences Dept., Univ. of Wisconsin-Madisonj Feb. 1990

[18] Jayant R. Haritsa, et al., Value-Based Scheduling in Real-Time Database Systems, Fred J. Maryanski, Editor, VLDB Journag2 117-152 (1993)

[19] JIA XU and DAVID LORGE PARNAS, Scheduling Processes with Release Times, Deadlines, Precedence, and Exclusion Relations, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 16. NO. 3. MARCH 1990

[20] Jiandong Huang e.tal," On Using Priority Inheritance in Real-Time Databases", National Science Foundation under Grant IRI-8908693 and Grant DCR-8500332, and by the U.S. O ce of Naval Research under Grant N00014-85-K0398., March, 1991

[21] K.Ramamrithm& et. Al., Efficient Scheduling Algorithm for Real Time Multiprocessor systems, COINS Technical Report 89-37, April,1989

[22] Kam-yiu Lam & et. al., Performance Studies of Locking Protocols for Real-time Databases with Earliest Deadline First, ***Journal of Database Management,*** *Vol. 6 No. 2,* Manuscript originally submitted November 10, 1993; Revised August 10, 1994; Accepted September 1, 1994 for publication.

[23] Kung, H. and J. Robinson, "On Optimistic Methods for Concurrency Control," ACM Trans. on Data-base Syst., vol. 6, no. 2, pp 213-226, June 1981.

[24] L. Sha, R. Rajkumar, and J. Lehoczky, "Concurrency Control for Distributed Real-Time Databases," ACM SIGMOD Record, vol. 17, no. 1, Mar. 1988.

[25] ÖzgürUlusoy&Bilkent, Lock-Based Concurrency Control in Distributed Real-Time Database Systems, ***Journal of Database Management,*** Manuscript originally submitted November 10, 1992; Accepted March 3, 1993 Accepted publication March 3, 1993

[26] P. Yu, K. Wu, K. Lin, and S. H. Son, "On Real-Time Databases: Concurrency Control and Scheduling," Proceedings of IEEE, vol. 82, no. 1, January 1994, pp 140-157.

[27] Philip S. & et. al., "On Real-Time Databases: Concurrency Control and Scheduling1994

[28] Rajendran M. et al., Priority assignment in real-time active databases, The VLDB Journal (1996) 5: 19–34

[29] Ranjana Jhala1 et al., Concurrency Control Model for Distributed Database, International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056, Volume: 02 Issue: 01 | Mar-2015

[30] Robinson, J. Design of concurrency controls for transaction processing systems, Ph.D. Thesis, Carnegie-Mellon University, Pittsburgh, PA, 1982.

[31] S. C. Cheng, et al.: Scheduling Algorithms for Hard Real-Time Systems | A Brief Survey. Hard Real-Time Systems, 150-173, IEEE (1988)

[32] Sang H. Son and Seog Park, Scheduling and Concurrency Control for Real-Time Database Systems, ONR, by DOE, by IBM, and, by CIT. 1992

[33] Sang H. Son, & et. al., Real-Time Database Scheduling: Design, Implementation, and Performance Evaluation, Database Systems for Advanced Applications'91. Ed. A. Makinouchi @World Scientific Publishing Co.,1991

[34] SHA, L., RAJKUMAR, R., AND LEHOCZKY, J. P. Priority inheritance protocols: An approach to real-time synchronization, CMU-CS-87-181, Dept. of Computer Science, Carnegie-Mellon Univ., Dec. 1987.

[35] Shiby et. al., Integrating Standard Transactions in Firm Real-Time Database Systems.1996

[36] Sonia Takkar. B.Tech, Scheduling Real-Time Transactions in paralle1 Database Systems, Carleton university Ottawa. Ontario. Iugust 1. 1997

[37] ulHaque, Waqar, "Transaction processing in real-time database systems " *Retrospective Theses and Dissertations.* (1993). Paper 10442.

[38] Yumnam Jayanta& Yumnam Somananda, Conflicting Management of Transactions in Real Time Database System, 2011 First International Conference on Informatics and Computational Intelligence,2011