An Overview of Adaptive Disk Scheduler for Real Time Database System

SALIM Y. AMDANI, PRASHANT A. BHALGE, SOHEL A. BHURA, SUREKHA B CHAVHAN

***Abstract*— The design and implementation of real-time database presents many new challenging problems. Compared with conventional database, real-time database have distinct features: they must maintain the coherent data while satisfy the timing constraints associated with transaction. In addition, real-time database must adapt to changes in operating environment and guarantee the completion of critical transaction with favoring changes in the system. With evolution of Earliest Deadline First (EDF) in 1973 by LIU and LAYLAND, laid the path for development of RTDB, it is very inefficient in overloaded workload conditions. Adaptive Earliest Deadline (AED) improves the performance which uses feedback control mechanism to detect overloaded condition and tries to attain HIT ratio 1.0. There prevails the risk of losing transaction with extremely high value may cause severe losses to system. A extension of AED called Hierarchical Earliest Deadline (HED) provide solution by establishing the value based bucket hierarchy thus ensuring the completion of high value transaction, in which value assigned reflects the return expected to receive if the transaction commits before its deadline. A new I-AED algorithm is studied which is based on EDF and AED Algorithm. A new multi-dynamic priority real-time scheduling algorithm named MDTS is studied, it considerates various characteristic parameters of transactions, and hard and soft real-time transactions are treated differently. This paper gives the overview of adaptive disk scheduling algorithms that can be used for real-time system.**

***Index Terms* -EDF, AED, HED,I-AED, MDTS, Real-Time, Overloaded**

I. INTRODUCTION

Real-time system manages their data in application dependent structures. As real-time systems evolve, their applications become more complex and require accessing more data. It thus becomes necessary to manage the data in more systematic and organized manner. Database management system provides tools for such organization, so in recent year there has been interest in "merging" database and real-time system. The resulting integrated system which provides the database operations with real-time constraint is called as real-time database system (RTDBS) as in [14]. Like conventional database real-time database act as the pool of data, provides the efficient storage, and performs the retrieval and manipulation of information. However, as a part of real-time system, whose "task" is associated with the time constraints, a RTDBS has an additional burden of imposing the time constraint to the "transaction".

A Real-Time Database System (RTDBS) is a transaction processing system that is designed to handle transactions with the timing constraint. Several previous

RTDBS as in [14] studies had been done to address the issue of scheduling transactions with the objective of minimizing the number of miss transaction. A common observation of these studies has been that, if we assigning priorities to transactions according to an Earliest Deadline policy minimize the number of miss transactions in systems operating under low or moderate levels of workload condition. But it fails in heavy loaded workload condition Hence, a question arises, how to improve the performance in overloaded workload conditions, There was a need of algorithm which would respond to the different condition and completing the transactions with favouring changes in the system. Adaptive earliest deadline first (AED) is the priority assignment algorithm which stabilize the overloaded conditions. It uses cost-effective feedback control mechanism to achieve performance guarantees in unpredictable environments. The primary goal of the RTDBS is to maximize the value realized by the in-time transactions and minimizing the number of miss transaction in the system is secondary concern. Hierarchical earliest Deadline (HED) is an algorithm which uses both values and deadline characteristic of the transaction to schedule them[6][7].

The paper is organized as follows: The next section contains the background materials i.e. discusses the EDF and AED with their working explained by algorithm. This section also includes value based scheduling using HED algorithm. A new algorithm I-AED i.e. Improved Adaptive Earliest Deadline which uses the transaction parameter to determine the workload condition.

The new algorithm based on Multi-dynamic Transaction Scheduling Algorithm. In this algorithm transactions are separated into three levels by type, that is, hard real-time transaction (HT), soft real-time transaction (ST), non-real-time transaction (NT). Their priorities are defined as: Prio (HT)>Prio (ST)>Prio (NT). Then, different kinds of transactions use different scheduling policies to assign priorities. Lowest priority to the non-real time transaction .Here one transaction is consider for NT.EDF scheduling assigns the highest priority to the transactions which have the earliest deadline, LSF scheduling assigns the highest priority to the shortest slack time transactions.

This paper gives the overview of adaptive disk scheduling algorithms that can be used for real-time system.

Manuscript received December 20, 2013.

Prof. S. Y. Amdani, Assoc. Professor and Head, Dept. of CSE B. N. C. O. E, Pusad (India) 9764996786.
Prof. P. A. Bhalge Assis. Professor , Dept. of CSE B. N. C. O. E, Pusad (India) 8087879738
Prof. S. A. Bhura, Assoc. Professor Dept. of CSE B. N. C. O. E, Pusad (India) 9764996766.
Miss. Surekha B. Chavhan M.E. Student, Dept. of CSE, B. N.C. O. E, Pusad (India) 9404236015

II. BACKGROUND

A. Disk Scheduling Problem

In a disk-based database system, disk I/O occupies a major portion of transaction execution time. As with CPU scheduling, disk scheduling algorithms that take into account timing constraints can significantly improve the real-time performance. CPU scheduling algorithms, like Earliest Deadline First and Highest Priority First, are attractive candidates but have to be modified before they can be applied to I/O scheduling. The main reason is that disk seeks time, which accounts for a very significant fraction of disk access latency, depends on the disk head movement. The order in which I/O requests are serviced, therefore, has an immense impact on the response time and throughput of the I/O subsystem[17].

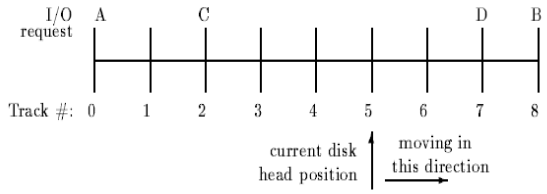

Figure1- Disk Scheduling Example

III. OVERVIEW OF EXISTING ALGORITHM

A. Earliest Deadline First(EDF)

In 1973 Liu and Layland, suggested the most popular real time disk scheduling algorithm Earliest Deadline First EDF[11]. The Earliest Deadline First algorithm is an analog of FCFS. Requests are ordered according to deadline and the request with the earliest deadline is serviced first. Assigning priorities to transactions an Earliest Deadline policy minimizes the number of late transactions in systems operating under low or moderate levels of resource and data contention. The EDF only consider the order of deadlines and introduces huge amount of seek-time costs with poor disk throughput [11].

B. AED Algorithm

Motivation for development of AED was the disadvantage of EDF algorithm. The Adaptive Earliest Deadline priority assignment algorithm modifies the Earliest Deadline policy based on following observation: When a set of transactions with deadlines that can all somehow be met, an Earliest Deadline policy will *also* meet all the deadlines. The significance of this observation is that in order to maximize the number of transactions that could meet their deadline, an Earliest Deadline schedule should be used among the largest set of transactions that can all be completed by their deadlines. The AED as in [6] algorithm uses a feedback control process to estimate the number of transactions that are sustainable under an EDF schedule.

Group Assignment: In AED algorithm, transaction executing in the system are collectively divided into groups, **HIT group** and **MISS group** as in [6]. On arrival of transactions in the system are assigned to a group based on unique integer key I_T which is assigned to newly arrived transaction randomly. Then these quantised transactions are arranged into key ordered list and position of each transaction posT **is** noted. If ***post*** is less than or equal to **HITcapacity** which dynamic control variable of AED is assigned to HIT group else to MISS group.

Priority Assignment: After assigning the new transactions to group, priority for each transaction is calculated using the following formula:

1) Priority in HIT group:

$$Pt = (1, Dt, It) = 1 + (1/Dt)$$

2) Priority in MISS group:

$$Pt = (0, Dt, It) = (1/Dt)$$

With this priority assignment scheme, transactions in HIT group will always have priority higher than transaction MISS group. To schedule the transactions in the HIT group EDF policy is used and Random Priority is used to schedule the transactions in the MISS group. IT component of priority serves to break the tie for the transaction in the HIT group having same deadline. Priority assigned to these transactions remains intact to them till they are in the system.

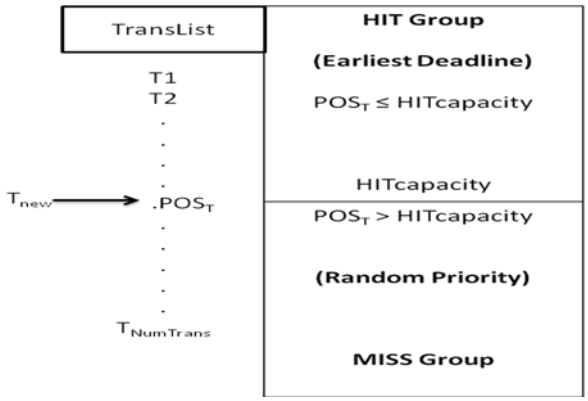

Figure 2-AED Priority Mapping

Goal of AED: Here main aim of the goal of the AED algorithm is to collect the largest set of transactions that can be completed before their deadlines in the HITgroup. It tries to achieve this by controlling the size of the HITgroup, using the HITcapacitysetting as the control variable. Transactions that cannot be accommodated in the HITgroup are considered to miss their deadlines and are therefore are assigned to the MISSgroup. **HIT Ratio** of a transaction group is fraction of transaction that had completed the execution before its deadline.

Control Variable Computation (HITcapacity): HITcapacity provides AED, the capacity to adapt to different workload condition and stabilize over-loaded condition. Information obtained from the system output after execution of the transactions is provided as "feedback" to calculate the control variable i.e. HITcapacity. HITRatio(HIT) and HITRatio(ALL) are the feedback information received from the system output. HITRatio(HIT) is the fraction of transactions in Hit group is making their deadline and HITRatio(ALL) is fraction of transaction in measure for all transactions in the system. Using these information and TransNum used to denote the number of transactions in the system; we can calculate the HITcapacity using following steps:

IF (HITRatio(ALL) < 0.95) then

HITcapacity = MIN (HITcapacity, HITRatio(ALL) * TransNum * 1.25)
ELSE
HITcapacity = HITRatio(HIT) * HITcapacity * 1.05

CASE 1: Feedback process is utilized to establish the control variable. The new HITcapacityis evaluated based on hit ratio observed in the HIT group; the size of the HIT group is adaptively changed to achieve a 1.0 hit ratio. Goal of AED is not just to have a HitRatio(HIT) of 1.0, but to achieve this goal with the **largest**possible transaction population in the HIT group as in [6]Just for this reason a **5** percent expansion factor **(1.05)** is included in the else part. This expansion factor ensures that the HITcapacityincreases steadily until the number of transactions in the HIT group is large enough to generate a HitRatio(HIT) of **0.95**. Now the transaction population size in the HIT group is close to the required number, and the HITcapacityremains stabilized at this setting (since 0.95 * 1.05~1.0).
CASE 2: A special care has to be taken while computing the HITcapacity: If the system experiences a prolong period of HitRatio(ALL) close to 1.0 due to the system being lightly loaded, HitRatio(HIT) will be 1.0 over this extended time period. In this situation, the HITcapacity can become very large due to the 5 percent expansion factor integrated while calculating, that is, there is a "runaway" effect. If the transaction arrival rate now increases such that the system becomes overloaded which is signalled by HitRatio(ALL) falling below 0.95, bringing the HITcapacity down from its artificially high value to the optimal level could take a considerable large amount of time. This will cause the system to enter in the unstable high-miss region of Earliest Deadline as every new transaction will be assigned to the HIT group due to the high HITcapacity setting. To prevent this from occurring, an upper bound on the HITcapacity value is used in to deal with the transition from a lightly-loaded condition to an overloaded condition. The upper bound is set to be 25 percent greater than an estimate of the "optimal" HitCapacity value, which is derived by computing the number of transactions that are currently making their deadlines. After the HITcapacity is quickly brought down in this fashion to near the appropriate setting.

Feedback Control Process (FCP)
FCP architecture in AED algorithm is consist of a control loop composed of monitor, controller, quality actuator and basic scheduler.
Feedback Control Loop:

1. Monitor measures the system output information requires to compute the control variable.

2. Controller uses these feedbacks information to compute control variable. The Controller uses a control function to compute the correct manipulated variable value which is used to compensate for the load variations and keep the controlled variables close to the optimal requirement.

3. Quality Actuator dynamically compares the position of the transactions entering in the system to assign them the respective group. Quality actuator has only two quality levels HIT group and MISS group.

4. Basic Scheduler: The FCP architecture has a Basic Scheduler that schedules admitted tasks with a scheduling policy (e.g., EDF and Random Priority). The properties of the scheduling policy can have significant impact on the design of the feedback control loop. This FCS architecture permits plugging in different policies for this Basic Scheduler and then designing the entire feedback control scheduling system around this choice.

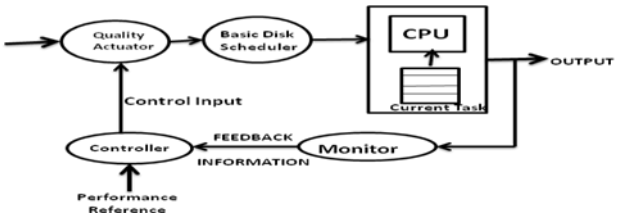

Figure3- For Feedback Control Loop

Control Related Variables: An important step in designing the FCS architecture is to decide the following variables of a real-time System in terms of control theory.

Controlled variables as in [6][7] are the performance metrics controlled by the scheduler. In feedback control process control variable is set which is used to divide the transactions into groups. FCP uses two parameters HITbatch and ALLbatch to compute the control variable. Transactions assigned in HIT section are marked with a special label HITbatch. At the RTDBS output, the status of completion (HIT or MISS transactions) of these specially-marked transactions is monitored. When the last transaction of HITbatch exits the system, HitRatio(HIT) is measured as the fraction of these transactions that completed before their deadline. The HitRatio(ALL) is continuously measured at the output as the hit ratio of the last transaction in ALLbatch that exited from the system. Where ALLbatch is label assigned to transactions in system. After each measurement of HitRatio (HIT), the HitRatio(HIT) value is fed back to the controller along with the current HitRatio(ALL) value. The controller then revaluates the HITcapacity, after which the whole process is repeated.

C. *Hierarchical Earliest Deadline (HED)*

This algorithm had considered the case where transactions have different values assigned to them. The goal here is to maximize the sum of the values of those transactions that commit by their deadline, and minimizing the number of missed deadlines becomes a secondary concern. A fundamental problem when transactions are characterized by both value and deadline is how to construct a priority ordering. It was found that one of two mappings − either Earliest Deadline (ED) or Highest Value (HV), which implement extreme tradeoffs − almost always provided the best performance.

Hierarchical Earliest Deadline (HED) as in [6] is extension of the AED algorithm which adaptively varies the trade off between value and deadline to maximize the value realized by the system. Informally, the HED algorithm groups transactions, based on their values, into a hierarchy of prioritized buckets. It then uses an AED-like algorithm within each bucket to determine the relative priority of transactions belonging to the bucket.

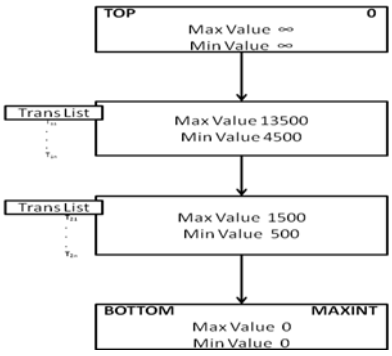

Figure4- HED Bucket Hierarchy

Bucket Assignment: The HED algorithm operates in the following manner: The *priority mapper* unit maintains a *value-based* dynamic list of buckets. Every transaction, upon arrival, is assigned based on its value to a particular bucket in this list. Each bucket in the list has an associated *MinValue* and *MaxValue* attribute − these attributes bound the values that transactions assigned to the bucket may have. Each bucket also has an identifier, and bucket identifiers in the list are in monotonically increasing order. There are two special buckets, TOP and BOTTOM that are always at the head and tail of the list, respectively.

The *MinValue* and *MaxValue* attributes of TOP are set to ∞, while the *MinValue* and *MaxValue* attributes of BOTTOM are set to zero. Since we assume that all transaction values are finite and positive, no transactions are ever assigned to these buckets, and their function is merely to serve as permanent list boundaries. The identifiers of the TOP and BOTTOM buckets are preset to 0 and MAXINT, respectively. When a new transaction, *Tnew*, arrives in the system, it is assigned to the bucket closest to TOP that satisfies the constraint $MinValue \leq Valuenew \leq MaxValue$. If no such bucket exists, a new bucket is inserted in the list between the bucket closest to TOP that satisfies $MinValue < Valuenew$ and its predecessor, and the transaction is assigned to this bucket. A newly created bucket is assigned its identifier by halving the sum of the identifiers of its predecessor and successor buckets. The *MinValue* and *MaxValue* attributes of a bucket are set as follows: Each bucket maintains an *AvgValue* attribute that monitors the average value of the set of transactions that are *currently* assigned to the bucket. The *MinValue* and *MaxValue* attributes of the bucket are then computed as (*AvgValue/SpreadFactor*) and (*AvgValue*SpreadFactor*), respectively, where *SpreadFact or* is a parameter of the HED algorithm. The *SpreadFactor* parameter controls the maximum spread of values allowed within a bucket. Whenever a transaction enters or leaves the system, the associated bucket updates its *AvgValue*, *MinValue* and *MaxValue* attributes. Hence after bucket assignment the transactions within each bucket are scheduled using AED like algorithm.

D. Improved Adaptive earliest deadline (I-AED)

The I-AED will use a method which will focus on using analysis of arrival-time of transaction, seek-time and transaction size to determine best scheduling algorithm for the current workload, switching and tuning algorithm as necessary to improve performance. This algorithm will use to determine the overloaded, under-loaded condition at the time when transactions will enter in the system and utilize the algorithms accordingly.

Overload Condition:

A real-time system is a processing system that is designed to handle workloads whose transactions have completion deadlines. The objective of the real-time system is to meet these deadlines; that is, to process tasks before their deadlines expire. Therefore, in contrast to the conventional computer systems where the goal is on satisfying the timing constraints of tasks. Under ideal condition, transactions never miss deadlines and this behaviour would be as expected. In reality, however, unanticipated emergency conditions may occur where the processing required to handle the emergency exceeds the system capacity, thereby resulting in missed deadlines[3][6]. The system is then said to be in **overload workload condition.** If this happens, it is important that the performance of the system degrades.

Parameter Related to I-AED:

I Parameters considered

Parameter	**Description**
Tid	Unique ID assigned to transaction request
It	Random Integer value assigned to transaction individually
TranSize	Transaction Size ie. No. of blocks in the transaction
AvgExt	Average execution time of the transaction
St	Seek time
TTT	Total transaction time
T.a	Request time or arrival time
T.e	Execution Time
T.d	Deadline Assigned to transaction
T.s	Slack Factor
T.er	Remaining time for execution of transaction
Du	Disk Utilization Factor
Feasibility%	Percentage Ratio of Feasible transaction in system
Mean-Du	Mean of Disk Utilization Factor

Feasibility Test of Transaction:

In this model, each input transaction T is independent of all other transaction and is completely characterized by three attributes:

1) T.a = the request time
2) T.e = the execution requirement
3) T.d = the relative deadline, often called as the deadline

The significance of these parameters is that transaction T, for successful completion, needs to be allocated the processor for **T.e** units of time during the interval **[T.a, T.a + T.d)**.

We assume that the system has knowledge of transaction parameters only at the instance when it makes the service request at time (T.a). Transactions that complete execution

by their deadlines are of value to the user application; that is, all deadlines are firm.
Slack Factor = T.s = T.d/T.e
Slack factor of transaction T is defined to be the ratio T.d/T.e and is denoted by T.s; it is a quantitative indicator of the tightness or slackness of the transaction deadline. It is niggling to see that it is necessary T.s ≥ 1 for it to be at all possible to complete a transaction before its deadline. In this study, we consider transaction sets where it is known a priori that all transaction in the transaction set will have a slack factor of at least f, where f≥1 is a calculated constant.

A task T is said to be active at time-instant t if:
1) It has requested service by time t (i.e., T.a≤ t),
2) Its service is not complete (i.e., T.er > 0, where T.er is the transaction's remaining service requirement), and
3) Its deadline has not expired (i.e., t < T.a + T.d).
An active task T is feasible at time t if T.er ≤ (T.a + T.d - t); that is, it is still possible to meet the task's deadline.

Disk Utilization Factor:
This factor is used to indicate the ratio of time for which the disk is busy in seeking to access the desired data as compare to accessing the required data.
EXAMPLE: If the current position of head is at location 30 and has to access the transaction of block size 5 is present at the location 3456, then time required by disk head to reach to desired block at location 3456 from location 30 will very large as compare to the time required for accessing the transaction block.
Hence, in a system if majority of transaction posses same condition then large amount of time is require for seeking as compare to accessing and processing of transaction. So there is chance of increase in number of transaction which will miss their deadline, resulting to overload workload condition.
Disk utilization Factor is denoted by: Du

$$Du = \frac{seek\ time}{average\ execution\ time}$$

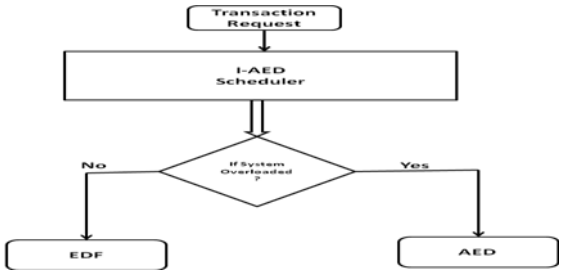

Figure 1- I-AED Scheduling Model

I-AED scheduler as explained in the figure-5 uses the result of feasibility test of transactions in the system and disk utilization factor and determines the workload condition i.e. under-load and overload condition. I-AED algorithm initially tests the feasibility of all transactions in the system and computes their disk utilization factors, is used to calculate the Feasibility percentage and mean disk utilization factor. In fifth step of the algorithm, if Feasibility percentage less than 95 percent, then system is declared to be overloaded and use AED algorithm. Else it check for Mean-Disk-Utilization factor if it is greater than 1 then system is declared overloaded then use AED algorithm. If above both condition are not satisfied then system is said to be under-loaded and EDF is used to schedule the transactions.

E. MDTS (Multi-dynamic Transaction Scheduling Algorithm)

Yuehua,Jing Proposed new algorithm MDTS in 2010. It considerates various characteristic parameters of transactions, and hard and soft real-time transactions are treated differently.Priority allocation is a key issue in transactionscheduling algorithm. It is affectted by many factors, suchas resource requirements, urgency degree, timing constraintsand so on. According to transaction's type, It designs amulti-dynamic priority assignment policy using deadlineand slack time. MDTS uses different methods to assignpriorities for different types of transactions.Transactions are sperated into three levels by type,thatis,hard real-time transaction (HT), soft real-time transaction(ST), non-real-time transaction (NT). Their priorities are defined as: Prio (HT)> Prio (ST)> Prio (NT). Then,different kinds of transactions use different schedulingpolicies to assign priorities. The priority of the transaction isultimately reflected into the priority of the process ofoperating system[19]. Therefore, the realization of the prioritypolicy needs to combine with embedded real-time operatingsystem process priority. In μC / OS-II, the process can bedivided into 64 levels (0 ~ 63), the higher the priority, thesmaller the number, the system takes up 8 priorities, that is0, 1,2,3, 60,61, 62, 63. Non-real-time transactions are set tobe the lowest priority level (defined as 59). In the electricpower control system, Hard real-time transactions are muchless than soft real-time transactions, so we set hard real-timetransactions priorities range 4 ~ 19, while soft real-timetransactions priorities range 20 ~ 58. Thus, when a newtransaction arrives, it can be assigned to correspondingpriority according to transaction type.Hard real-time transaction's priority assignmentcombines EDF and LSF algorithms, using deadline D andslack time S,these two factors to decide .From the LSFalgorithm, It's known that the slack time of preemptivedynamic scheduling is defined as:S = d_e-(t_0 + E-P);de the deadline of transaction, t0 the current time, E P,respectively mean estimated time of the implementation ofthe transaction T and elapsed runtime, S dynamicallychanges over time. Meanwhile, From EDF algorithm ,therelative deadline is defined as:D = de-t0;according to the definition of the slack time, because of theremainder of transaction execution time is greater than zero,therefore, A hard real-time transaction is meaning toschedule only when its slack time is shorter than the relativedeadline, or have the necessary to discuss their priorities[19].Therefore, when we calculate the priority of the hard realtimetransactions T, we use α the weight of the relativedeadline D and the slack time S to insure these two factors,and then use map_ht function to map into the correspondinghard real-time transaction priority, that is,priority (T) = map_ht (α * D + (1-α) * S).In addition,In order to ensure the consistency of priorities,when we calculate the priority of a new transaction, It'sneed to dynamically adjust their priorities which have the same type but higher priorities.Soft real-time transactions' priorities are according toLSF algorithm, and the slack time is defined as:S = de-(t0 + E-P);priority allocation function:priority (T) = map_st (S);map_st function is used to map the different times of softreal-time transaction to corresponding priority.

IV. CONCLUSION

Thus, we have studied various real time transaction scheduling algorithms like EDF, AED, HED and a new algorithm i.e. I-AED which is based on EDF and AED. In EDF transactions are ordered according to deadline and the request with earliest deadline is serviced first. But EDF fails in the overloaded condition .To avoid this, AED algorithm is proposed and it performs better than EDF. It works on adaptive method. On hit group it applies EDF and on miss group it applies random priority algorithm. Furthermore, an enhanced AED algorithm called the hierarchical AED is proposed in a manner in which it obtains a better packet serving performance by using the concept of priority based on quality of service (QOS) of network traffic rather than using a random priority assignment when doing group assignment. A new multi-dynamic priority real-time scheduling algorithm named MDTS is studied, it considerates various characteristic parameters of transactions, and hard and soft real-time transactions are treated differently.

REFERENCES

[1] Abbott, A. and H. Garcia-Molina, "Scheduling Real-time Transactions: a Performance Evaluation", Proceedings of the 14th VLDB Conference, 1988.

[2] Abbott, A. and H. Garcia-Molina, "Scheduling Real-time Transactions with Disk Resident Data", Proceedings of the 15th VLDB Conference, 1989.

[3] Sanjoy K. Baruah,andJayant R. Haritsa "Sheduling for Overload in Real-Time Systems" IEEE TRANSACTIONS ON COMPUTERS, VOL. 46, NO. 9, pp. 1034-1039, SEPTEMBER 1997

[4] ErdoganDogdu ,GultekinOzsoyoglu, "Real-Time Transactions with Execution Histories: Priority Assignment and Load Control." Department of Computer Engineering and Science Case Western Reserve University Cleveland, OH 44106.National Science Foundation grant IRI-92-24660.

[5] M. Dorigo and T.Stutzle, Ant Colony Optimization , The MIT Press,Cambridge, 2004.

[6] Jayant R. Haritsa, MironLivny, Michael J. Carey, " Earliest Deadline Scheduling for Real-Time Database Systems"National Science foundatiaou under grantIRI-8657323.

[7] Haritsa, J.. "Transaction Scheduling in Real-Time Database Systems," PhD. Thesis**.** Computer Sciences Department, Univ. of Wisconsin Madison August 1991.

[8] Huang et al, "Experimental Evaluation of Real-Time Transaction processing." Proc. of 10th IEEE Real-Time Systems Symposium. Dec. 1989

[9] Ben Kao and Hector Garcia-Molina, "Ovierview of Real-Time Database System" PrinctonUniversity,Princton NJ 08544 USA,Standford University,Standford,CA,94305,USA(1991)

[10] KetanKotecha and Apurva Shah, "Adaptive Scheduling Algorithm for Real-Time Operating System" 978-1-4244-1823-7/08/$25.00 _2008 IEEE.

[11] C. Liu, J.W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," *Journal. ACM*, pp. 104-112,Jan. 1973.

[12] Alma Riska, Erik Riedel, Sami Iren "Adaptive disk scheduling for overload management" Proceedings of the First International Conference on the Quantitative Evaluation of Systems (QEST'04) 0-7695-2185-1/04 $ 20.00 IEEE.

[13] Sang H. Son, Prasad Wagle, and Seog Park, "Real-Time Database Scheduling: Design, Implementation, and Performance Evaluation" DATABASE SYSTEMS FOR ADVANCED APPLICATIONS '91 Ed. A. Makinouchi @World Scientific Publishing Co.

[14] Saud A. Aldarmi, "Real-Time Database Systems: Concepts and Design"Department of Computer Science The University of York April 1998

[15] ShenodaGuirguis, Mohamed A. Sharaf, PanosK.Chrysanthis, AlexandrosLabrinidis, Kirk Pruhs "Adaptive Scheduling of Web Transactions." IEEE International Conference on Data Engineering. 1084-4627/09 $25.00 © 2009 IEEE DOI 10.1109/ICDE.2009.137

[16] S.Y.Amdani "Comparison of Earliest Deadline with Adaptive Earliest Deadline Algorithm" International Conference on Global Technology Initiatives vol. no.1 ISBN 978-93-5067-450-5

[17] S.Y. Amdani, M.S. Ali "An Overview of Real-Time Disk Scheduling Algorithms." *International Journal on Emerging Technologies* **2**(1): 126-130(2011)

[18] LimeiTang ,Suxia Xing ,Tianhua Chen, "An improved adaptive cache prefetch algorithm" 2012 Fifth International Symposium on Computational Intelligence and Design. 978-0-7695-4811-1/12 $26.00 © 2012 IEEE DOI 10.1109/ISCID.2012.215

[19] Zhao Yuehua, Qiu Jing "A new multi-dynamic priority real-time database scheduling algorithm."978-1-4244-6349-7/10/$26.00 _c 2010 IEEE