# Transaction Scheduling in Multiclass Real-Time Database Systems

HweeHwa Pang
Miron Livny
Michael J. Carey

Technical Report #1110

September 1992

# Transaction Scheduling in Multiclass Real-Time Database Systems†

*HweeHwa Pang*
*Miron Livny*
*Michael J. Carey*

Computer Sciences Department
University of Wisconsin - Madison

## ABSTRACT

In real-time database systems (RTDBS), the policy used for assigning priorities to transactions may lead to a biased treatment of different transaction classes. Such bias is not necessarily a reflection of differences in the tightness of the time constraints of these classes. Instead, it may be due to discrimination in the allocation of system resources. While bias due to differences in time constraint tightness is arguably reasonable, discrimination based on other class features (e.g., mean transaction size) may be unacceptable in many real-time applications.

In this study, we focus on firm RTDBSs with multiclass workloads where classes are distinguished by their transaction sizes. Our results reveal that the Earliest Deadline scheduling principle, upon which a number of existing priority assignment policies are based, discriminates significantly against longer transactions. This observation has motivated the development of a new dynamic priority assignment scheme that improves the chances for long transactions to meet their time constraints, thereby providing a fairer mechanism for use in multiclass RTDBS transaction scheduling.

† A slightly abridged version of this paper that will appear in the proceedings of the 13th Real-Time Systems Symposium, December 1992.

## 1. INTRODUCTION

In real-time database systems (RTDBS), priorities are used to translate time constraints into scheduling decisions. Most researchers who have studied RTDBS algorithms have adopted the Earliest Deadline (ED) principle for priority assignment in their performance studies [e.g., Abbo89, Huan91, Hari91]. The ED principle was first introduced by Liu and Layland in [Liu73] and has been extensively analyzed and evaluated since then [Dert74, Mok78, Jens85, Panw88]. The results of these studies have shown that ED generates schedules that miss very few deadlines, if any, when the system is operating in a region where it should theoretically be able to meet all or most of the time constraints. This paper focuses on how the ED policy treats different transaction classes when the time constraints cannot all be satisfied.

When assigning priorities to transactions, ED considers only their deadlines; the closer the deadline, the higher the priority. In this way, ED attempts to maximize the number of transactions that finish on time, ignoring the issue of which particular transactions meet their deadlines and which fail to do so. However, such an approach may lead to problems in systems with multiclass workloads. In particular, one class of transactions may miss a significant fraction of their deadlines while another transaction class misses none. As in any timesharing system, differences in the performance provided to different transaction classes cannot simply be ignored, as fairness is also an important performance consideration.

In the context of general-purpose timesharing systems, the processor-sharing (PS) scheduling policy is generally viewed as treating jobs in a fair manner. In particular, the PS policy ensures that the expected waiting time per unit of service requested will be the same for all job classes [Klei76]. Based on similar reasoning, it can be argued that the probability of a transaction meeting its deadline should be a function only of the tightness of its time constraint, and not of other class attributes. Thus, it is reasonable to expect that two classes of transactions with similarly tight time constraints will have comparable miss ratios. The results of our study, however, show that an RTDBS that uses an ED-based scheduling policy is unlikely to meet this expectation. In fact, as we will see in the next section, such a policy discriminates against longer transactions in attempting to minimize the number of late transactions.

To overcome the bias of existing ED-based scheduling approaches, this paper introduces a dynamic priority assignment policy, called *Adaptive Earliest Virtual Deadline* (AEVD), that considers both the time constraints and the arrival times of transactions. By means of a sequence of virtual deadlines, AEVD attempts to ensure that long transactions are allocated a fair share of the system resources. The AEVD policy is designed for use in a firm RTDBS environment, where transactions that miss their deadlines are aborted and discarded. The general architecture of a firm

RTDBS is depicted in Figure 1. Each incoming transaction is assigned a priority by the *priority mapper*, the component responsible for implementing the priority assignment policy of the system. The assigned priority is then used by the different priority-conscious schedulers of the system. A feedback loop enables the priority mapper to continuously monitor the performance of the system and to dynamically adjust its priority assignment policy when necessary.

The only other studies that we are aware of that have addressed the issue of fairness in the context of real-time systems are presented in [Zhao87] and [Huan91]. Zhao and Ramamritham showed in [Zhao87] that, in the context of hard real-time communication, ED achieves a small ratio of message loss by biasing towards short messages. They used a workload in which messages of all sizes had the same average slack time. Thus long messages had relatively tighter deadlines than short messages. This could conceivably have been responsible for the bias observed there. The results of our study will show that even if messages of all sizes were given equally tight deadlines, a similar biased behavior would still have been observed, hence confirming that ED is biased.

In [Huan91], Huang et al briefly examined fair real-time scheduling issues from a concurrency control perspective. A two-class workload was used to show that, as in conventional database systems, optimistic concurrency control in an RTDBS discriminates against classes that contain longer transactions. By assigning an appropriate weighting factor to each class, it was shown that priorities can be used to overcome this concurrency control bias.

The remainder of this paper is organized as follows: Section 2 illustrates the biased behavior of the Earliest Deadline scheduling policy in multiclass RTDBSs. The AEVD algorithm is introduced in Section 3. A detailed simulator of a firm RTDBS, intended for studying the performance of the AEVD algorithm, is described in Section 4. Section 5 details the results of a series of simulation experiments showing that, over a wide range of multiclass workloads, the new policy allocates system resources in a fair way, and at the same time significantly reducing the fraction of the

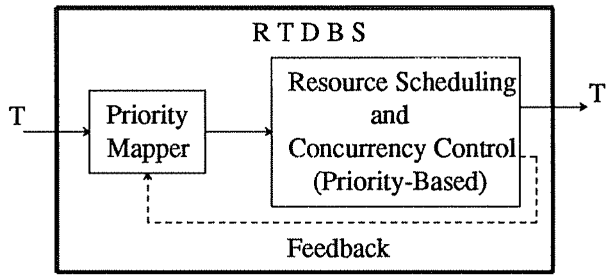
RTDBS
T
Priority Mapper
Resource Scheduling and Concurrency Control (Priority-Based)
T
Feedback

Figure 1: RTDBS Architecture

offered load that is not completed on time. Finally, our conclusions are presented in Section 6.

## 2. THE BIASED BEHAVIOR OF EARLIEST DEADLINE

The Earliest Deadline (ED) algorithm is based on a simple but effective principle: requests with closer deadlines should be served before requests whose deadlines are further away. Unlike priority assignment algorithms that assume a priori knowledge of the resource requirements of each request, the ED algorithm uses only information that is readily available in any real-time system — it equates the priority of a request to its deadline. This property of the ED algorithm makes it a very attractive candidate for real-time database systems where the resource requirements of transactions are unlikely to be known in advance.

ED has been shown to produce, for a wide range of RTDBSs, high success rates under light load conditions [Abbo88, Abbo89, Hari90a, Huan89, Huan91, Kim91]. However, the performance of ED deteriorates rapidly when the system is overloaded and cannot meet the deadlines of a significant fraction of the transactions. An Adaptive Earliest Deadline (AED) algorithm that stabilizes the overload performance of ED was introduced in [Hari91]. The AED algorithm augments the Earliest Deadline principle with a process that randomly predicts the destiny of a transaction when it arrives at the system. Depending on the current miss ratio of the system, the new transaction is assigned to one of two groups: the "hit" group or the "miss" group. The larger the current miss ratio, the higher the chances that the new transaction will be placed in the "miss" group. AED then uses the Earliest Deadline policy to order the execution of transactions in the "hit" group; transactions in the "miss" group get to use a system resource only when there are no transactions from the "hit" group at the same resource. This guarantees that transactions from the "miss" group have almost no impact on the progress of transactions from the "hit" group.

Both ED and AED do not take into account transaction arrival times in their priority assignment decisions. Thus, they give transactions no credit for arriving early and waiting for resources. This leads to a situation where short transactions that arrive just before their deadlines have an advantage over long transactions that arrive way before their deadlines. Table 1 is a quantitative display of the advantage that short transactions have over long transactions under these two policies. It presents the performance of three transaction classes, I, II, and III, in a simulated RTDBS which is described in Section 4. The transaction sizes, expressed in terms of the number of pages accessed, of these classes are uniformly distributed in the range 1 to 10, 11 to 20, and 21 to 30, respectively. The time constraint of a given transaction, which is the difference between the transaction's deadline and its arrival time, varies uniformly between 2 and

|  | Miss Ratio |  |  | Relative Priority |  |  |
|---|---|---|---|---|---|---|
| **Class** | **ED** | **AED** | **NP** | **ED** | **AED** | **NP** |
| *I* | 0% | 4% | 26% | 0.80 | 0.78 | 0.50 |
| *II* | 6% | 7% | 25% | 0.45 | 0.45 | 0.50 |
| *III* | 36% | 22% | 25% | 0.25 | 0.27 | 0.50 |
| System | 14% | 11% | 25% |  |  |  |

Table 1: Class Performance

6 times the size of the transaction. The first column of Table 1 shows a very significant difference between the miss ratios of the three classes under ED. While ED misses none of the class I transactions, it misses more than a third of the transactions of class III. Since on the average all transaction classes have equally tight time constraints, it can be argued that ED discriminates unfairly against class III transactions, treating class I transactions more favorably. AED exhibits a similar, though less skewed, behavior. It misses 4% of the class I transactions and "only" 22% of the class III transactions.

Under the No Priority (NP) policy, also shown in Table 1, the distribution of miss ratios is very different. NP assigns the same priority to all transactions and therefore each resource services requests in the order that they arrive (First Come First Serve). As a result, the miss ratio distribution for NP reflects the relative tightness of the time constraints of transactions, which is the same in this case, rather than their relative sizes. As expected, the system miss ratio of NP is much larger than the system miss ratios of the other two policies. These results show that while NP treats the three classes fairly but not efficiently, ED and AED are efficient but unfair.

In order to understand why ED and AED discriminate against longer transactions while NP treats the three transaction classes fairly, we have to examine the relative priority of the three classes under the different policies. The relative priority of a class captures the average rank of its transactions relative to transactions of other classes over their lifetime in the system; the rank of a transaction that has the highest priority in the system is one, while the rank of a transaction that has the lowest priority is zero. When the average rank of two transactions differ, the one with the higher average rank has precedence to system resources. Thus it will progress more steadily and so have a better chance to meet its deadline. It is therefore expected that when a policy discriminates against a class of transactions, the relative priority of this class will be smaller than the relative priority of a class that receives favorable treatment. The relative priority of the three classes for the different algorithms, also shown in Table 1, confirm this expectation. Whereas all three classes have the same relative priority under NP, shorter transaction classes have higher relative priorities for ED and AED.

The cause of the skew in the relative priority of the three classes under ED can be illustrated by the following example: Consider a long and a short transaction that have the same deadline and whose time constraints are equally tight. The long transaction arrives before the short transaction and will receive a low rank at the beginning because its deadline is far away. The rank of the long transaction will then increase gradually as its deadline approaches. Once the short transaction joins the system the two transactions will have the same rank. Since the average rank of a transaction is accumulated over its lifetime in the system, the short transaction has a higher average rank than the long one. Therefore, overall, the short transaction will make more steady progress towards meeting its deadline.

A more detailed view of the biased behavior of ED and AED is provided by Figure 2, where transaction miss ratios are presented as a function of transaction size for the simulations underlying Table 1. The figure clearly shows that, while NP treats transactions of all sizes equally, ED and AED make longer transactions pay the price for improving the performance of shorter transactions. Under the two ED-based algorithms, miss ratio is unmistakably an increasing function of transaction size; In other words, the longer a transaction is, the less likely it will make its deadline. Moreover, there is a distinctive miss ratio discrepancy between transactions of size 1 and size 30. In fact, under ED more than half of the transactions of size 30 miss their deadlines, while none of the transactions of size 1 are late. The corresponding miss ratios under AED are 34% and 3%, respectively. A detailed analysis of the behavior of ED and AED can be found in Section 5. Here it suffices for us to note that while the ED principle is simple and produces high success rates, it can be extremely biased. It is therefore not likely to be suitable for use in multiclass RTDBSs.

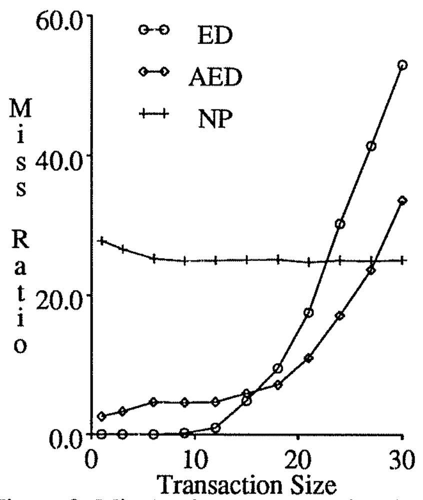

Figure 2: Miss Ratio vs. Transaction Size

## 3. ADAPTIVE EARLIEST VIRTUAL DEADLINE (AEVD)

The goal of the *Adaptive Earliest Virtual Deadline* (AEVD) algorithm is to preserve the high-success-rate virtue of the Earliest Deadline principle while avoiding its biased behavior. This algorithm is designed for multiclass workloads where classes are distinguished by their mean sizes and the time constraints of transactions are, on the average, proportional to transaction size. In addition to deadlines, AEVD uses transaction arrival times in its priority assignment decisions so that transactions get credit for arriving early and waiting for resources.

Throughout the remainder of this paper we use $D_T$, $A_T$, and $P_T$ to denote the deadline, arrival time, and priority of transaction $T$, respectively. Priority assignments are such that smaller $P_T$ values reflect higher priorities. In addition, $C_T = D_T - A_T$ is used to denote the time constraint of $T$. These notations are summarized in Table 2, together with some notations which we will introduce later.

AEVD is based on the Adaptive Earliest Deadline (AED) algorithm [Hari91]. Before AEVD is introduced, we review the main features of AED. In AED, all of the transactions that are currently active are divided into a "hit" group and a "miss" group. Upon arrival, each transaction is randomly assigned a unique key, $I_T$. The transaction is then inserted into a *key-ordered* list of transactions, and its position in the list, $POS_T$, is noted. If $POS_T$ is less than *HITcapacity* (the capacity of the "hit" group), the new transaction is assigned to the "hit" group; otherwise it gets assigned to the "miss" group. *HITcapacity* is dynamically adjusted so that the miss ratio of the "hit" group is less than 5%. The adjustment is done after every *HITbatch* transactions leave the "hit" group, where *HITbatch* is an algorithm parameter. The following scheme is used to assign priorities to transactions:

$$P_T = \begin{cases} (0, D_T, I_T) & \text{if } Group = hit \\ (1, 0, I_T) & \text{if } Group = miss \end{cases}$$

This scheme defines a lexicographical priority order, giving all transactions in the "hit" group higher priorities than transactions in the "miss" group. Priority ordering among transactions in the "hit" group is by Earliest Deadline, while Random Priority (RP) is used for the "miss" group. Under RP, the priority of an incoming transaction is randomly selected

| Notation | Meaning | Notation | Meaning |
|---|---|---|---|
| $A_T$ | Arrival time | $SD_T$ | Service Demand |
| $C_T$ | Time Constraint | $SR_T$ | Slack Ratio |
| $D_T$ | Deadline | $P_T$ | Priority |
| $Size_T$ | Transaction Size | $PF_T$ | Pace factor |

Table 2: Notations

from a pre-defined range of values and remains fixed throughout its lifetime.

The AEVD algorithm differs from AED in that it uses an *Earliest Virtual Deadline* (EVD) policy, instead of Earliest Deadline, to manage the "hit" group. The remainder of this section is devoted to a description of EVD.

## 3.1. Earliest Virtual Deadline

The EVD policy uses a sequence of virtual deadlines to control the pace at which a transaction progresses towards meeting its deadline. The relative distances between these virtual deadlines reflect what the EVD policy considers as a desired pace for the given transaction. Under this policy, the virtual deadline, $V_T$, replaces the actual deadline of the transaction, $D_T$, as the tie breaker in the "hit" group:

$$P_T = \begin{cases} (0,\, V_T,\, I_T) & \text{if } Group = hit \\ (1,\, 0,\, I_T) & \text{if } Group = miss \end{cases}$$

Every time a transaction requests one of the system resources, its current virtual deadline is tested and, if it has expired, a new virtual deadline is assigned:

$$V_T = (D_T - Clock) \times PF_T + Clock$$

where $0.0 < PF_T \leq 1.0$ is the *pace factor* of the transaction. Examples of several transactions and their virtual deadlines are presented in Figure 3. We will come back to these examples later.

EVD assigns a pace factor to every transaction when it arrives at the system, and this factor remains fixed throughout its lifetime. The pace at which a transaction advances towards its deadline is determined by its pace factor. When a transaction first arrives, EVD assigns it a virtual deadline of $C_T \times PF_T + A_T$. The intention in selecting this point in time as the first virtual deadline is for the transaction to complete $PF_T$ of its work by the time $PF_T$ of its time constraint has passed. Once the current virtual deadline of the transaction expires, this rule is applied recursively on its residual time constraint to place the next virtual deadline. Therefore, the smaller $PF_T$ is, the more regular the transaction's progress will be. By controlling the pace factors assigned to transactions, EVD can maintain their

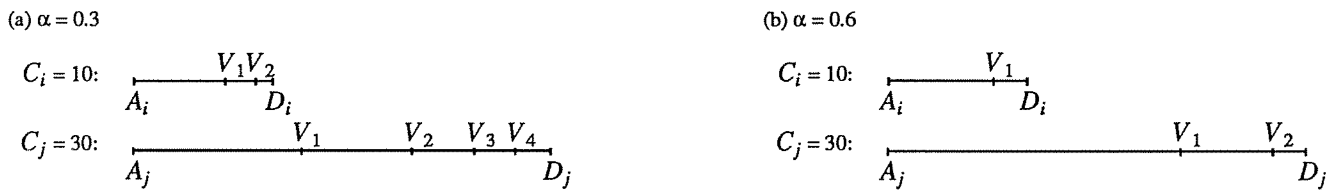

Figure 3: Virtual Deadline Assignment

progress rates at a desired level.

To overcome ED's discriminatory behavior, EVD needs to monitor the progress of longer transactions more closely to ensure that they are advancing steadily towards meeting their deadlines. In other words, longer transactions require smaller pace factors. Since an RTDBS does not usually have a priori knowledge of the processing requirements of incoming transactions, the size of transactions has to be estimated. Assuming that the time constraint for a transaction is at least as large as its total service demands, it is not unreasonable to expect that a transaction's time constraint will provide some indication of its size. Moreover, a look at a wide range of synthetic workloads that have been used in prior RTDBS performance studies reveals that time constraints and transaction sizes are indeed positively correlated, i.e., longer transactions tend to have longer time constraints. (Since we do not have any actual real-time database workload as yet, these synthetic workloads provide the best indication of the kind of workloads that we can expect in practice.) EVD therefore determines the pace factor of a transaction from its time constraint in the following way:

$$PF_T = \alpha + (1 - \alpha) \times \left(\frac{C_{\max} - C_T}{C_{\max} - C_{\min}}\right)^2$$

where $\alpha$ is a control variable of the algorithm and $C_{\min}$ and $C_{\max}$ are, respectively, the minimum and maximum time constraints of the last $2 \times$ *HITbatch* transactions[1]. The shape of $PF_T$ as a function of $C_T$ is shown in Figure 4.

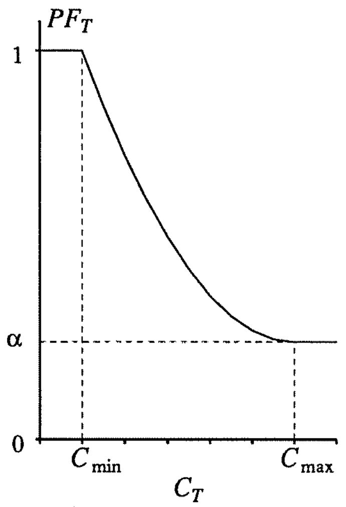

Figure 4: Pace Factors

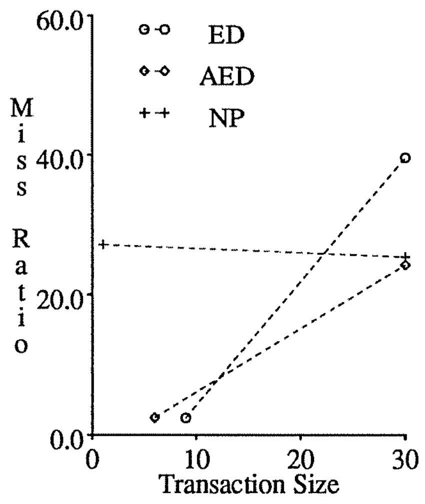

Figure 5: Regression Lines

[1] $PF_T = 1.0$ if $C_T < C_{\min}$, and $PF_T = \alpha$ if $C_T > C_{\max}$.

The dynamic variable $\alpha$ determines the range for the pace factor. Transactions whose time constraints are equal to $C_{\min}$ always have a pace factor of one, which implies that each of these transactions has only one virtual deadline (which is equal to its actual deadline). The pace factor of a transaction with a time constraint of $C_{\max}$ is $1 - \alpha$. A smaller $\alpha$ will therefore result in transactions with long time constraints (and indirectly long transactions) receiving smaller pace factors as compared to transactions with short time constraints. This is illustrated by the examples in Figure 3, which presents the virtual deadlines assigned to two transactions, one of size 10 and the other of size 30, using $\alpha$ values of 0.3 and 0.6. By assigning a "suitable" value to $\alpha$, then, EVD can control the progress rate and hence the relative success rate of long and short transactions.

The EVD algorithm uses a feedback mechanism to monitor the performance of the system, and then adjusts the value of $\alpha$ accordingly[2]. The feedback is in the form of a *bias factor*, denoted by *BF*, that measures the linear correlation [Triv82] between miss ratio and transaction size. *BF* is equal to the slope of the regression line that best fits the miss ratios expressed as a function of transaction size. Thus *BF* captures the bias of the underlying priority assignment algorithm. A positive *BF* is an indication that longer transactions have higher miss ratios, which suggests that there is a bias against longer transactions. Conversely, a negative *BF* signifies that the underlying algorithm is favoring longer transactions. As an illustration, the regression lines resulting from the miss ratio curves in Figure 2 are shown in Figure 5. The figure shows that for this experiment ED, with a *BF* of 1.77, is the most biased. AED is second with a *BF* of 0.91. NP treats all transactions fairly and produces a *BF* of -0.05.

In order to dynamically determine an appropriate value for $\alpha$, the *BF* value is updated each time the capacity of the "hit" group is recomputed. After that, $\alpha$ is either reduced or increased by 5%, subject to a lower limit of 0.01 and an upper limit of one[3]. If miss ratio and transaction size are positively correlated, i.e. $BF > 0$, $\alpha$ is reduced; if $BF < 0$, is increased. Note that if $\alpha = 1.00$, every transaction receives only one virtual deadline, and EVD degenerates to ED.

## 4. REAL-TIME DATABASE SYSTEM MODEL

The RTDBS performance model used in this study is essentially the same as that in [Hari90a]. We summarize the model here for completeness and to aid the reader in interpreting the results.

[2] $\alpha$ is initialized to 0.5 at system startup time.

[3] By definition, $\alpha$ has to be greater than 0.00 and is at most equal to 1.00. To prevent $\alpha$ from becoming fixed at zero, we require $\alpha \geq 0.01$.

The RTDBS consists of a shared-memory multiprocessor that operates on disk-resident data. The database is modeled as a collection of pages. A transaction consists of a sequence of read and write page accesses. A read access involves a concurrency control request to get access permission, a disk I/O to read the page, followed by a period of CPU usage for processing the page. Write requests are handled similarly, except that their I/O activities are deferred until the transaction has committed. A transaction that gets restarted due to a data conflict follows the same access pattern as the original transaction. Transactions that are not completed by their deadlines are aborted and discarded.

The structure of the RTDBS performance model is shown in Figure 6. There are five major components: a *Source* that generates transactions in a Poisson stream and assigns a deadline to each transaction; a *Transaction Manager* that models the execution of transactions, and implements the priority assignment algorithm; a *Concurrency Control Manager* that implements the concurrency control algorithm used; a *Resource Manager* that models the CPU and disk resources; and a *Sink* that gathers statistics on completed transactions.

## 4.1. Resource Model

The physical resources in our model consist of multiple CPUs and disks. There is a single queue for the CPUs and the service discipline is preemptive-resume based on transaction priorities. Each of the disks has its own queue and uses a non-preemptive priority scheduling policy. The top half of Table 3 summarizes the key parameters of the resource model. The *NumCPUs* and *NumDisks* parameters specify the system's resource composition, and the *PageCPU* and *PageDisk* parameters are constants that capture the CPU and disk processing times per data page. The service time per page is defined as the sum of *PageCPU* and *PageDisk.* The data is assumed to be uniformly distri-

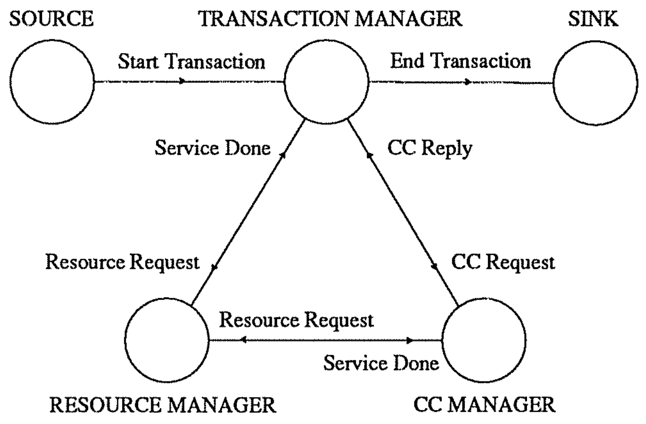

Figure 6: RTDBS Model Structure

buted across all of the disks.

## 4.2. Workload Model

The workload model characterizes transactions in terms of the number of pages that they access and their deadlines. Let the size of a transaction $T$, $Size_T$, denote the number of pages accessed by $T$. The service demand of $T$, $SD_T$, is defined as the product of $Size_T$ and the total service time per page. The *Source* module assigns a deadline to each new transaction in the following manner:

$$D_T = SD_T \times SR_T + A_T$$

where $D_T$, $SR_T$ and $A_T$ are the deadline, slack ratio and arrival time of $T$, respectively. Recall that the time constraint of $T$, $C_T$, equals $D_T$ - $A_T$, or equivalently, $SD_T \times SR_T$. $SR_T$ determines the tightness/looseness of the deadline of $T$.

The bottom half of Table 3 summarizes the key parameters of the workload model. The *ArrivalRate* parameter specifies the mean transaction arrival rate. *DatabaseSize* gives the number of pages in the database. Each page that is read will be updated with probability *WriteProb*. The number of pages accessed by a transaction varies uniformly in the range specified by *SizeInterval*; page requests are generated from a uniform distribution (without replacement) spanning the entire database. The slack ratio of a transaction, $SR_T$, varies uniformly in the range specified by *SRInterval*.

## 5. EXPERIMENTS AND RESULTS

## 5.1. Performance Metrics

Two primary performance metrics are used in this paper: *Normalized Miss Ratio* and the bias factor *BF*. *Normalized Miss Ratio* captures the fraction of the offered load that is not completed on time. It is computed as:

| Parameter | Meaning |
|---|---|
| *NumCPUs* | Number of CPUs |
| *NumDisks* | Number of disks |
| *PageCPU* | CPU time for processing a data page |
| *PageDisk* | Disk service time for each data page |
| *ArrivalRate* | Transaction arrival rate |
| *DatabaseSize* | Number of pages in the database |
| *WriteProb* | Write probability per accessed page |
| *SizeInterval* | Range of the number of pages accessed per transaction |
| *SRInterval* | Range of slack ratio |

Table 3: Input Parameters

$$Normalized\ Miss\ Ratio = \frac{\sum_{s \in SizeInterval} s \times MissRatio_s}{\sum_{s \in SizeInterval} s}$$

where $MissRatio_s$ denotes the miss ratio of transactions of size $s$, and *SizeInterval* is the range of transaction sizes in the workload. We chose this metric because unlike *average miss ratio*, which can be made artificially low by servicing only shorter transactions (which require less resources), *Normalized Miss Ratio* is an *unbiased* performance measure for multiclass systems. To illustrate this point, consider the experiment discussed in Section 2 again. The *average miss ratios* for ED, AED and NP were, respectively, 14%, 11% and 25%. This gives a misleading impression that ED and AED perform significantly better than NP. The corresponding *Normalized Miss Ratios* are 23%, 15% and 25%, which indicates that ED actually misses almost as much of the offered load as NP! In terms of *Normalized Miss Ratios*, AED is still better than NP, but the difference in performance is much less than what is indicated by the *average miss ratios*. Note that the choice of *Normalized Miss Ratio* over *average miss ratio* in multiclass RTDBSs is similar to the preference for *page throughput* over *transaction throughput* in conventional multiclass database systems [Care90].

The second performance metric is the bias factor, denoted by *BF*, which captures the linear correlation between miss ratio and transaction size. The *BF* metric was introduced in Section 3.

## 5.2. Overview of Experiments

A simulator based on the RTDBS model described in Section 4 is used to evaluate the performance of the priority assignment algorithms. The simulator is written in DeNet [Livn90], a simulation language based on Modula-2. In the simulator, the mean CPU time to process a page is set to 10 milliseconds, while the mean disk access time is set to 20 milliseconds. The number of CPUs and disks are set to 8 and 16, respectively. *HITbatch*, a parameter of both AED and AEVD, is set to 60. Recall that the capacity of the "hit" group is re-evaluated after every *HITbatch* transactions leave the "hit" group. We begin with a baseline model, and then further experiments are carried out by varying a few parameters each time.

## 5.3. Baseline Model

The focus of this model is on the effect of resource contention on the performance of the various algorithms, therefore the parameter *WriteProb* is set to zero. As a result, there is no data contention. We will introduce data contention in a later experiment to study the combined effect of both forms of contention.

Table 4 summarizes the workload and resource parameter settings for the baseline model. In the model, transaction size varies uniformly between 1 and 30, and slack ratio distribution is independent of transaction size. The system does not have any a priori knowledge of transaction resource requirements, so late transactions can only be discarded after they have missed their deadlines. A system with *Normalized Miss Ratios* in the range 0%-25% is considered to be moderately loaded, while *Normalized Miss Ratios* greater than 25% indicate heavy load conditions.

We first examine the performance of the algorithms under moderate load conditions by setting the transaction arrival rate to 47.5 transactions/second. (This system condition was used to generate the results in Section 2.) Figure 7 plots miss ratio versus transaction size for the baseline model. The solid lines in the figure show the miss ratios produced by the various algorithms, and the dotted lines are the corresponding regression lines. The figure indicates a marked difference between the behavior of AEVD and that of ED and AED: AEVD's *BF* of 0.00 implies that it does not manifest any biased behavior. In contrast, ED and AED obviously discriminate against longer transactions — ED has a *BF* of 1.78, and AED has a *BF* of 0.91. AEVD's *Normalized Miss Ratio* is computed to be 16%, which is comparable to the 15% produced by AED and much lower than ED's 23%. This indicates that AEVD is able to produce

| Workload Parameter | Value | Resource Parameter | Value |
|---|---|---|---|
| *DatabaseSize* | 1000 pages | *NumCPUs* | 8 |
| *WriteProb* | 0.0 | *NumDisks* | 16 |
| *SizeInterval* | [1,30] | *PageCPU* | 10 ms |
| *SRInterval* | [2.0,6.0] | *PageDisk* | 20 ms |

Table 4: Parameter Settings for the Baseline Model

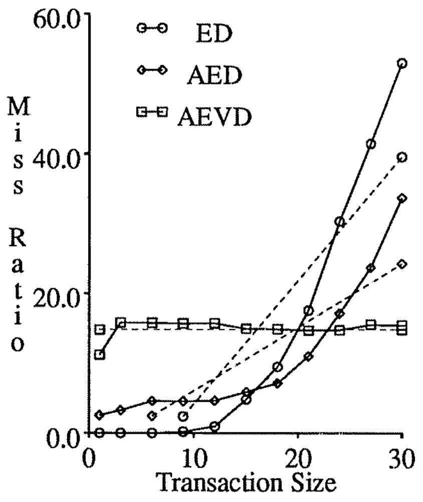

Figure 7: Baseline (λ = 47.5)

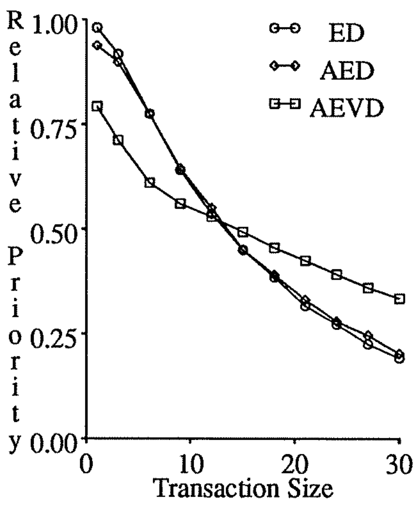

Figure 8: Relative Priorities

low miss ratios and at the same time remain unbiased.

To understand why ED and AED discriminate against longer transactions whereas AEVD is unbiased, we examine the relative priority of transactions of different sizes. The relative priority of a group of transactions is the average of the relative priority of all of its transactions, which is derived in the following way: at any instant, all the transactions in the system are ranked by their priority values such that the transaction with the lowest and the highest priority is at the head and the tail of the list, respectively. The instantaneous relative priority of a transaction is computed by dividing the position of the transaction in the list by the total number of active transactions. Note that the instantaneous relative priority of the transaction with the highest and lowest priorities are 1.0 and 0.0 respectively. By averaging the instantaneous priority of a transaction over its lifetime, we obtain its relative priority.

Figure 8 presents the relative priorities as a function of transaction size. The figure indicates that relative priority decreases with increasing transaction size. Moreover, ED and AED produce significant discrepancies between the relative priorities of the longest and the shortest transactions — a factor of 5.1 in the case of ED and 4.6 in the case of AED. This is the reason why miss ratio is an increasing function of transaction size under these two algorithms. Under AEVD, the use of virtual deadlines reduces the discrepancy in relative priorities to a factor of 2.4; in other words, the relative priority of transactions of size 1 is only 2.4 times higher than the relative priority of transactions of size 30. Compared to ED and AED, then, longer transactions advance much more steadily and complete earlier under AEVD, while at the same time, shorter transactions are slowed down. This is demonstrated by Figure 9, which traces the aver-

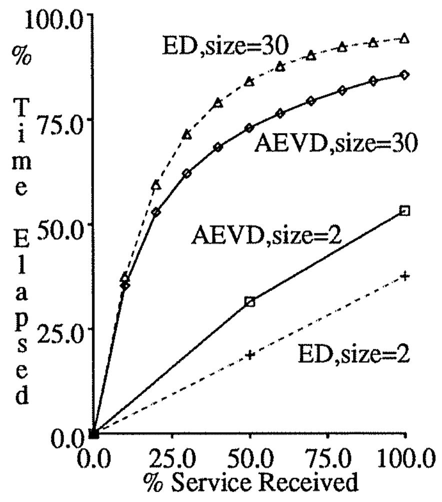

Figure 9: Rate of Progress

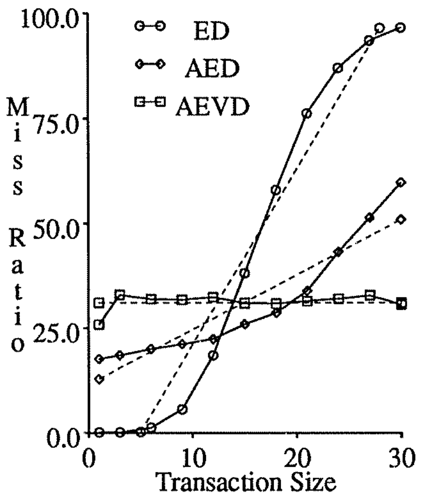

Figure 10: Baseline ($\lambda = 60.0$)

age progress of transactions of sizes 2 and 30 under ED and AEVD. (The corresponding AED curves, not shown in the figure, are similar to those of ED.) In the figure, *% Service Received* is defined as the ratio of service received to service demand, and *% Time Elapsed* is the ratio of the elapsed time to the time constraint. Under ED, transactions of size 30 require almost 95% of their time constraints to complete, while AEVD reduces this to 85%. The impact of AEVD on shorter transactions is also evident in Figure 9, which shows an increase in their completion times from 35% of their time constraints in the case of ED to 50% in the case of AEVD. The earlier completion of longer transactions is what lowers their miss ratios. At the same time, the later completion of shorter transactions causes their miss ratios to go up. Although long transactions still have a lower relative priority and take a higher percentage of their time constraints to complete than short transactions, the longer absolute time constraints associated with the long transactions make them less susceptible to the effect of transient overload conditions caused by bursty arrival patterns. This is the reason why AEVD is able to produce balanced miss ratios across transactions of all sizes.

Next, we increase the arrival rate from 47.5 to 60.0 transactions/second. This brings the operating conditions of the system to a heavy load. Figure 10 shows miss ratio as a function of transaction size. Comparing Figure 10 with Figure 7, we note that the discriminatory behavior of ED worsens considerably under heavy load. The almost 100% misses for transactions of size 30 and no misses for transactions of size 1 and 2 increase ED's *BF* from 1.78 to 4.19. The performance of AED does not deteriorate as much because the capacity of the "hit" group limits the effect of ED's biased behavior. However, we still witness an increase in its *BF* from 0.91 to 1.31. In contrast, AEVD performs as well under heavy loads as it does under moderate load conditions: its curve remains almost flat, and it has a *BF* of 0.00. Moreover, its *Normalized Miss Ratio* of 36% is slightly lower than AED's 38% and much lower than ED's 64%.

Having seen the *BF* values of ED, AED, and AEVD for arrival rates of 47.5 and 60.0, we present in Figure 11 the *BF* values for the various algorithms as a function of the arrival rate. The corresponding *Normalized Miss Ratios* are given in Figure 12. We shall analyze the behavior of each algorithm in turn.

Figure 11 shows that the bias of ED worsens considerably with increasing arrival rate until an arrival rate of 60.0 transactions/second. At this point, the miss ratio of the longest transactions approaches 100%, as we have seen previously. After that, the system starts to miss more of the shorter transactions. This accounts for the slight drop in ED's *BF* values. The high *Normalized Miss Ratios* of ED are caused by the poor performance of longer transactions.

Next, we examine AED. At low arrival rates, most of the transactions get assigned to the "hit" group, where there is a bias against longer transactions due to the Earliest Deadline policy. This causes the initial rise in the *BF*

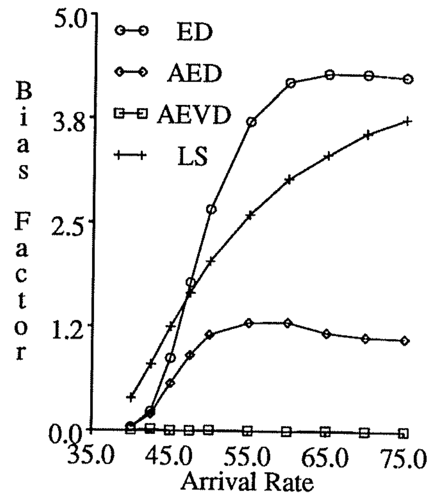

Figure 11: Baseline - *BF*

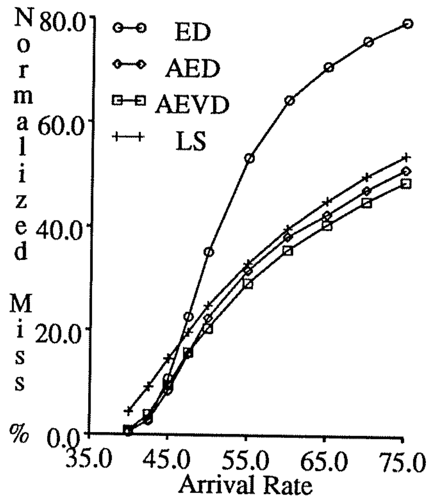

Figure 12: Baseline - Normalized Miss %

curve. At high arrival rates, however, a majority of the transactions get assigned to the "miss" group where Random Priority is used. Random Priority treats transactions of all sizes fairly, and produces low miss ratios under heavy loads [Hari91]. This is why AED's *BF* curve levels off. By combining the low-load characteristics of ED and the high-load behavior of RP, AED produces consistently low *Normalized Miss Ratios.*

We now turn our attention to AEVD. By using feedback on the performance of the system, AEVD is able to dynamically find appropriate $\alpha$ values for all of the load conditions, thus producing *BF* values that are close to zero. At the same time, AEVD has lower *Normalized Miss Ratios* than ED and AED. This is a consequence of its having a balanced mix of transaction sizes among the transactions that finish on time.

Since AEVD achieves balanced miss ratios by assigning priorities according to transaction sizes, it is interesting to compare AEVD with other algorithms that also consider transaction sizes. One such algorithm is *Least Slack* (LS). This algorithm computes the laxity of each transaction by subtracting its outstanding service demand from its remaining time constraint, and then gives the highest priority to the transaction that has the smallest laxity. The performance results of LS are included in Figures 11 and 12. These results show that, while the normalized miss % of LS is only slightly worse than those of AEVD, LS is biased against long transactions. This behavior can be explained as follows: The time that each request spends at a resource comprises both a waiting period and a service time. Long transactions, which are made up of many resource requests, are therefore more likely to have a longer total waiting period than short transactions. However, all transactions, regardless of size, gain enough priority to resources at the same average laxity under LS. Consequently, the total waiting period of longer transactions are more likely to exceed this average laxity, which causes these transactions to have higher miss ratios. Since LS is biased and requires knowledge of exact transaction sizes, it is not a promising candidate for use in database systems. We will therefore not consider LS further in this paper.

In summary, we can draw the following conclusions from our results for the baseline experiment. First, ED and AED discriminate significantly against longer transactions. Second, AEVD does not produce a noticeable biased behavior and has the lowest *Normalized Miss Ratio* among the algorithms that we considered. Hence, AEVD appears to be an attractive candidate for scheduling multiclass RTDBS workloads.

## 5.4. Transaction Size Distribution

In the baseline model, transactions of all sizes had the same arrival rate because transaction size varied uniformly between 1 and 30. The objective of this experiment is to examine how workloads with different transaction size distributions might affect the performance of the algorithms. We achieve that by modifying the transaction arrival process so that the probability that an incoming transaction has a size of *i* pages is inversely proportional to *i*; for example, an incoming transaction is 30 times as likely to have a size of 1 than a size of 30.

Figures 13 and 14 show the *BF* and *Normalized Miss Ratio* results, respectively, as a function of arrival rate. Since this workload has a larger proportion of shorter transactions, the load on the system is much lighter than in the case of the baseline experiment. Initially, ED and AED are able to meet almost all of the deadlines, and their *BF* values

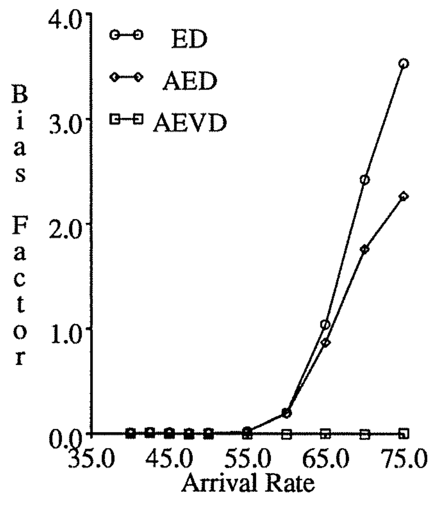

Figure 13: Size Dist. - *BF*

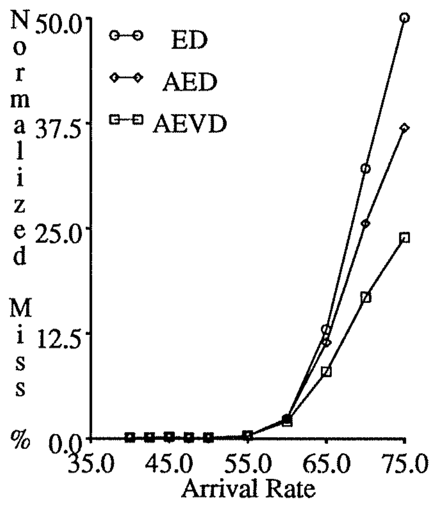

Figure 14: Size Dist. - Normalized Miss %

are therefore close to zero. At higher arrival rates where the system stands to miss deadlines, however, we again observe their biased behavior. There, the poor performance of longer transactions leads to high *Normalized Miss Ratios*. In contrast, AEVD produces balanced miss ratios, and its *Normalized Miss Ratios* are significantly lower than those of ED and AED under heavy loads.

We also experimented with different ranges of transaction sizes. The qualitative results of those experiments were the same as what we have seen here, which leads us to conclude that AEVD is robust under changes in the transaction size distribution.

## 5.5. Skewed Slack Ratio

AEVD was designed to operate with multiclass workloads where the time constraints of all transactions are, on the average, proportional to transaction size (i.e. where slack ratio distribution is independent of transaction size). In this experiment, we investigate the performance of AEVD for a workload where slack ratio and transaction size are correlated, making it more difficult to meet the deadlines of some classes of transactions than others.

For the first part of the experiment, we make the slack ratio proportional to size. Here, transactions with a size of 1 have a slack ratio of 2.0, and transactions with a size of 30 have a slack ratio of 6.0. Figures 15 and 16 show the resulting *BF* and *Normalized Miss Ratio* values. The looser time constraints of longer transactions give them an advantage over shorter transactions. At low loads, this advantage is sufficient to offset the effect of ED's biased behavior.

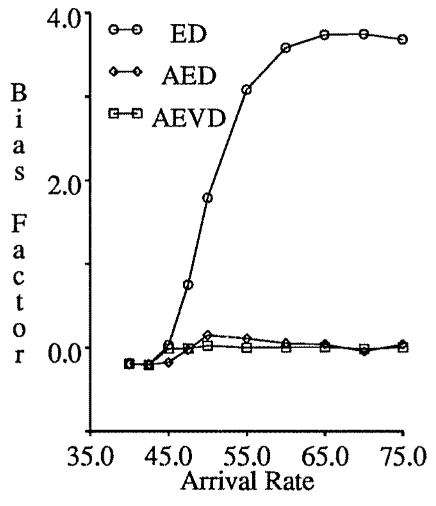

Figure 15: Positive Correlation - *BF*

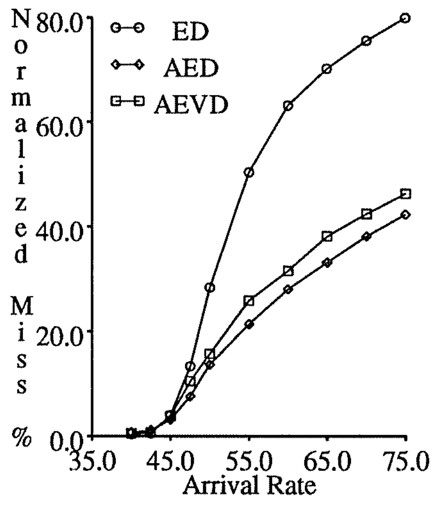

Figure 16: Positive Correlation - Norm. Miss %

Nonetheless, at high loads longer transactions still perform badly under ED. In contrast, AED has low *BF* values because the selection criteria for the "hit" group is independent of transaction size. Transactions in the "hit" group experience light-load conditions, where the discriminatory behavior of ED is compensated for by the advantage that longer transactions get from their looser deadlines. Therefore, the AED "hit" group has balanced miss ratios. We now examine AEVD. This policy adjusts for the advantage that longer transactions have by assigning fewer virtual deadlines to them, thereby causing transactions of all sizes to have balanced miss ratios and achieving *BF* = 0. For this workload, the use of virtual deadlines does not benefit longer transactions because their looser deadlines are already sufficient to help them overcome ED's discrimination. In fact, at high loads, competition from transactions that are not close to their deadlines causes the RTDBS to miss some urgent transactions which could have been met otherwise. This is the reason why Figure 16 indicates that AEVD has a slightly higher *Normalized Miss Ratio* than AED at high loads.

For the second part of the experiment, the correlation between slack ratio and size is reversed so that slack ratio is *inversely* proportional to size. Thus shorter transactions are given the looser deadlines, and longer transactions have tighter deadlines. Figures 17 and 18 show the *BF* and *Normalized Miss Ratio* results as a function of arrival rate. At low loads, ED and AED produce low *BF* values because most of the deadlines can be met. At high loads, however, the bias of ED again causes longer transactions to perform much worse than shorter transactions. In contrast, the high miss ratios of longer transactions in the "hit" group cause AED to limit the capacity of this group; as a result, most of the transactions end up being assigned to the "miss" group. This is why its *BF* curve levels off. We now examine AEVD.

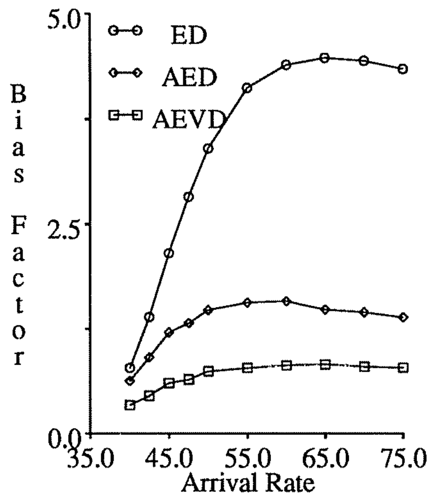

Figure 17: Negative Correlation - *BF*

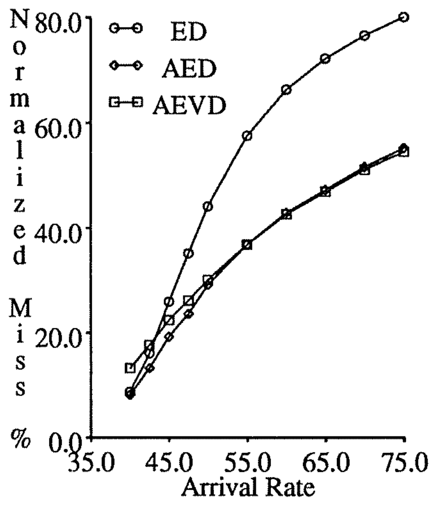

Figure 18: Negative Correlation - Norm. Miss %

By reducing the pace factors of longer transactions in the "hit" group to enable them to progress more steadily, AEVD helps them to overcome the handicap of their tighter deadlines. The miss ratios in the AEVD "hit" group are thus balanced. However, AEVD also attempts to maintain low *Normalized Miss Ratios* for the "hit" group, and this results in the group having a small capacity. Consequently, a substantial portion of transactions are assigned to the "miss" group where RP is used. Since longer transactions have deadlines that are harder to make, RP produces a positive *BF* in the "miss" group. This results in positive overall *BF* values, though the *BF* values are smaller than those produced by ED and AED. Another observation is that AEVD has higher *Normalized Miss Ratios* than AED at low loads, indicating that there is a performance penalty in making the system service the longer, tighter transactions when it could have completed more of the shorter, looser transactions with the same resources. At high loads, however, the performance gains from the longer transactions offset the loss in performance of the shorter transactions, so that the *Normalized Miss Ratios* for both AEVD and AED are about the same.

The results of this experiment show that if the slack ratio distribution makes the deadlines of some classes of transactions more difficult to meet than others, there can be a performance penalty in attempting to overcome the bias. Moreover, if we take into account the performance penalty involved, AEVD has the best overall performance among the algorithms that we have considered. To summarize, we can conclude that AEVD is fairly robust under variations in slack ratio distribution.

## 5.6. Data Contention

Our next experiment explores situations where both resource contention and data contention cause the system to miss deadlines. This is done by setting the write probability to 0.25, which means that one-fourth of the data items that are read will also be updated. Due to space limitations, we will only present results obtained for one concurrency control algorithm. Since previous studies have shown that optimistic algorithms outperform locking algorithms in firm RTDBSs under high data contention [Hari90b], we will use an optimistic concurrency control mechanism. The chosen algorithm is briefly described below.

In classical optimistic concurrency control (OPT) [Kung81], transactions are allowed to execute unhindered until they are ready to commit, at which time they are validated. A transaction is restarted if it fails its validation test, which checks that there is no conflict between the validating transaction and transactions that committed after it began execution. In the *Broadcast Commit* variant (OPT-BC) [Mena82, Robi82], a committing transaction notifies other currently running transactions that conflict with it so that these conflicting transactions can restart immediately. This method detects conflicts earlier than the OPT algorithm, and results in earlier restarts and less wasted resources. Performance studies have shown that OPT-BC produces low miss ratios in firm RTDBS under heavy loads [Hari90b]. We therefore choose this algorithm for our experiment.

Figures 19 and 20 present the performance results for this experiment as a function of the transaction arrival rate. Figure 19 shows the *BF* values, and Figure 20 gives the *Normalized Miss Ratios*. Among the three algorithms, the miss ratios produced by AEVD are the least sensitive to transaction size. Interestingly, at low arrival rates where data contention dominates resource contention, longer transactions are discriminated against even in the case of AEVD. In this situation, performance is primarily determined by the concurrency control algorithm, and OPT-BC is known to favor shorter transactions [Huan91]. At high loads, AEVD reduces the effect of the bias caused by OPT-BC by giving longer transactions higher priorities at the CPUs and disks. Both AEVD and AED have comparable *Normalized Miss Ratios*, while ED performs badly at high loads as expected.

The most important conclusion from this experiment is that, when data contention dominates resource contention, the use of priorities alone is not enough to overcome the bias caused by the concurrency control algorithm. We also ran experiments with other concurrency control algorithms, including 2PL [Eswa76, Gray79], 2PL-High Priority [Abbo88], OPT-WAIT [Hari90b], and WAIT50 [Hari90b]; these experiments revealed the same bias against longer transactions. To treat transactions of all sizes fairly under both data and resource contention, an unbiased concurrency control algo-

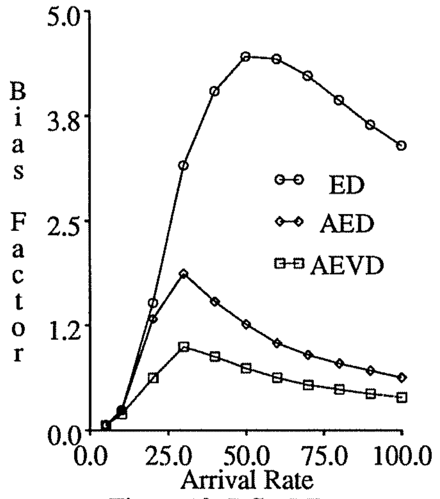

Figure 19: DC - *BF*

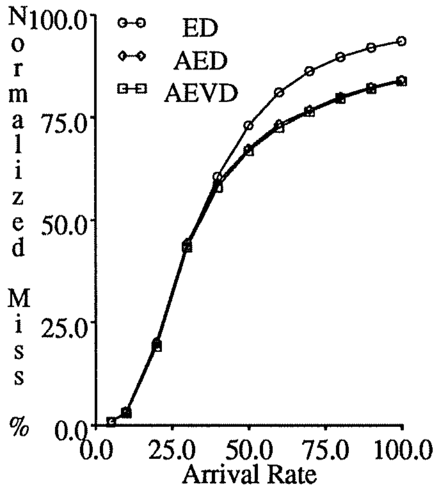

Figure 20: DC - Norm. Miss %

rithm that complements AEVD is required. Such an algorithm is now under study and will (hopefully) be the topic of a forthcoming paper.

## 5.7. Discussion

An issue that concerns the use of *BF* to represent the dependency of miss ratio on transaction size is: how good a representation *BF* really is? Recall that *BF* captures only the linear correlation between miss ratio and size. If there is a non-linear (e.g., curvilinear) correlation between miss ratio and size, we may still get a *BF* of zero. Thus, $BF = 0$ by itself does not allow us to conclude that miss ratio is independent of transaction size. As a safeguard, we also computed the standard error, $\sigma_{MissRatio}$, defined as

$$\sigma_{MissRatio} = [ \sum_{s \in SizeInterval} (MissRatio_s - y_s)^2 ]^{\frac{1}{2}}$$

where *SizeInterval* is the range of transaction sizes in the workload, and $y_s$ is the miss ratio of transactions of size $s$ predicted by the regression line. Denoting the mean miss ratio by E(MissRatio), the coefficient of variation is defined as $CV_{MissRatio} = \sigma_{MissRatio}$ / E(MissRatio). If we obtain small $CV_{MissRatio}$ values after fitting a regression line to the miss ratios produced by an algorithm, we know that the regression line models the miss ratios well. It then follows that the *BF* value is a good indicator of the correlation between miss ratio and size produced by the algorithm.

In our experiments, we observed that 70% of the time the $CV_{MissRatio}$ values obtained with AEVD were smaller than 0.15, and 90% of the time the $CV_{MissRatio}$ values were below 0.30. This shows that the regression lines modeled the behavior of AEVD fairly well. In the case of ED and AED, the $CV_{MissRatio}$ values were larger. However, regression lines still captured the general trends in the miss ratios that they produced. Regression lines therefore provide a simple and reasonably good indicator of the dependency of miss ratio on transaction size.

## 6. CONCLUSIONS

In this paper, we addressed the issue of priority assignment in firm multiclass real-time database systems (RTDBS) where classes are distinguished by their mean sizes. We showed that Earliest Deadline (ED) and Adaptive Earliest Deadline (AED) make longer transactions pay the price for reducing the overall miss ratio. In other words, there is a bias against longer transactions under these scheduling policies. This behavior may not be acceptable in many real-time applications. We introduced a new algorithm, called Adaptive Earliest Virtual Deadline (AEVD), that attempts to minimize the overall miss ratio without discriminating against longer transactions. AEVD divides the transactions in an RTDBS into a "hit" group and a "miss" group. Transactions in the "hit" group are given preferential access to resources to enhance the chances that they will make their deadlines. Moreover, the time constraint of each transaction in the "hit" group is divided into a series of *virtual* deadlines. At any time, the transaction with the earliest virtual deadline has the highest priority; by adjusting the tightness of the virtual deadlines assigned to transactions, the algorithm is able to control their progress and hence their miss ratios.

To quantify the behavior of the algorithms, we presented two performance metrics: *Normalized Miss Ratio*, which measures the fraction of the workload that is not completed on time, and the bias factor *BF*, which quantifies the linear correlation between miss ratio and transaction size. Using a detailed RTDBS simulation model, a series of experiments was carried out to study the performance of the algorithms. Our experiments revealed that, in attempting to ensure fair treatment, the system can incur a penalty in terms of the fraction of the workload that finishes on time. Our results also showed that AEVD outperformed ED and AED over a wide range of multiclass workloads. It achieved *BF* values that were close to zero, indicating that transactions of all sizes had balanced miss ratios. One exception was when shorter transactions had looser deadlines, making it difficult to lower the miss ratios of the longer, tighter transactions to achieve balanced miss ratios. Another exception was when data contention dominated resource contention, in which case the behavior of the concurrency control algorithm was the dominant performance factor. Even so, AEVD had the smallest *BF* values among all the algorithms that we considered in both cases. Moreover, with only one exception, AEVD also had the lowest *Normalized Miss Ratios*. The exception occurred when AEVD incurred a performance penalty for trying to improve the miss ratios of some classes of transactions that had tighter deadlines. In one particular instance, where the looseness of deadlines was proportional to transaction size, AED was also able to produce balanced miss ratios across transactions of all sizes, and at the same time produce lower *Normalized Miss Ratios* than AEVD at high loads. Overall, our results indicated that AEVD is an attractive alternative to ED and AED.

## REFERENCES

**[Abbo88]** R. Abbott, H. Garcia-Molina, "Scheduling Real-Time Transactions: A Performance Evaluation", *Proc. of the 14th Int. Conf. on Very Large Data Bases*, August 1988.

**[Abbo89]** R. Abbott, H. Garcia-Molina, "Scheduling Real-Time Transactions with Disk Resident Data", *Proc. of the 15th Int. Conf. on Very Large Data Bases*, August 1989.

**[Care90]** M. Carey, S. Krishnamurthi, M. Livny, "Load Control for Locking: The 'Half-and-Half' Approach", *Proc. of the 1990 ACM PODS Symposium*, April 1990.

**[Dert74]** M. Dertouzos, "Control Robotics: the procedural control of physical processes", *Proc. of IFIP Congress*, 1974.

**[Eswa76]** K. Eswaran, et al, "The Notions of Consistency and Predicate Locks in a Database System", *Communications of ACM*, November 1976.

**[Gray79]** J. Gray, "Notes on Database Operating Systems", in *Operating Systems: An Advanced Course*, R. Bayer, R.Graham, G. Seegmuller, eds., Springer-Verlag, 1979.

**[Hari90a]** J. Haritsa, M. Carey, M. Livny, "On Being Optimistic about Real-Time Constraints", *Proc. of the 1990 ACM PODS Symposium*, April 1990.

**[Hari90b]** J. Haritsa, M. Carey, M. Livny, "Dynamic Real-Time Optimistic Concurrency Control", *Proc. of the 1990 IEEE 11th Real-Time Systems Symposium (RTSS).*

**[Hari91]** J. Haritsa, M. Livny, M. Carey, "Earliest Deadline Scheduling for Real-Time Database Systems", *Proc. of the 1991 IEEE 12th Real-Time Systems Symposium (RTSS).*

**[Huan89]** J. Huang, J.A. Stankovic, D. Towsley, K. Ramamritham, "Experimental Evaluation of Real-Time Transaction Processing", *Proc. of the 1989 IEEE 10th Real-Time Systems Symposium (RTSS).*

**[Huan91]** J. Huang, J.A. Stankovic, D. Towsley, "Experimental Evaluation of Real-Time Optimistic Concurrency Control Schemes", *Proc. of the 17th Int. Conf. on Very Large Data Bases*, September 1991.

**[Jens85]** E. Jensen, C. Locke, H. Tokuda, "A Time-Driven Scheduling Model for Real-Time Operating Systems", *Proc. of the 1985 IEEE 6th Real-Time Systems Symposium (RTSS).*

**[Kim91]** W. Kim, J. Srivastava, "Enhancing Real-Time DBMS Performance with Multiversion Data and Priority Based Disk Scheduling", *Proc. of the 1991 IEEE 12th Real-Time Systems Symposium (RTSS).*

**[Klei76]** L. Kleinrock, *Queueing Systems, Volume II: Computer Applications*, John Wiley & Sons, Inc., 1976, pp. 166-170.

**[Kung81]** H. Kung, J. Robinson, "On Optimistic Methods for Concurrency Control", *ACM Transactions on Database Systems, Vol. 6,2*, June 1981.

**[Liu73]** C. Liu, J. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment", *Journal of ACM*, January 1973.

**[Livn90]** M. Livny, "DeNet User's Guide, Version 1.5", *Computer Sciences Department, University of Wisconsin - Madison*, 1990.

**[Mena82]** D. Menasce, T. Nakanishi, "Optimistic versus Pessimistic Concurrency Control Mechanisms in Database Management Systems", *Information Systems, Vol. 7,1*, 1982.

**[Mok78]** A. Mok, M. Dertouzos, "Multi-processor Scheduling in a Hard Real-Time Environment", *Proc. of the 7th Texas Conf. on Computing Systems*, October 1978.

**[Panw88]** S. Panwar, D. Towsley, "On the Optimality of the STE Rule for Multiple Server Queues that Serve Customers with Deadlines", *COINS Technical Report 88-81*, University of Massachusetts, Amherst, July 1988.

**[Robi82]** J. Robinson, "Design of Concurrency Controls for Transaction Processing Systems", *Ph.D. Thesis, Carnegie Mellon University*, 1982.

**[Triv82]** K. Trivedi, "Probability & Statistics with Reliability, Queuing, and Computer Science Applications", Prentice-Hall, Inc., Englewood Cliffs, NJ 07632, 1982, Chapter 11.

**[Zhao87]** W. Zhao, K. Ramamritham, "Virtual Time CSMA Protocols for Hard Real-Time Communication", *IEEE Transactions on Software Engineering, SE-13(8)*, August 1987.