Integrating User Preferences and Real-Time Workload in Information Services

Prabhudev Konana • Alok Gupta • Andrew B. Whinston
Department of MSIS, College and Graduate School of Business, The University of Texas at Austin, Austin, Texas 78712
Department of Operations and Information Management, 368 Fairfield Road, U-41 IM, University of Connecticut, Storrs, Connecticut 06269
Department of MSIS, College and Graduate School of Business, The University of Texas at Austin, Austin, Texas 78712
pkonana@mail.utexas.edu • alok@sba.uconn.edu • abw@uts.cc.utexas.edu

We propose priority pricing as an on-line adaptive resource scheduling mechanism to manage real-time databases within organizations. These databases provide timely information for delay sensitive users. The proposed approach allows diverse users to optimize their own objectives while collectively maximizing organizational benefits. We rely on economic principles to derive priority prices by modeling the fixed-capacity real-time database environment as an economic system. Each priority is associated with a price and a delay, and the price is the premium (congestion toll resulting from negative externalities) for accessing the database. At optimality, the prices are equal to the aggregate delay cost imposed on all other users of the database. These priority prices are used to control admission and to schedule user jobs in the database system. The database monitors the arrival processes and the state of the system, and incrementally adjusts the prices to regulate the flow. Because our model ignores the operational intricacies of the real-time databases (e.g., intermediate queues at the CPU and disks, memory size, etc.) to maintain analytical tractability, we evaluate the performance of our pricing approach through simulation. We evaluate the database performance using both the traditional real-time database performance metrics (e.g., the number of jobs serviced on time, average tardiness) and the economic benefits (e.g., benefits to the organization). The simulation results, under various database workload parameters, show that our priority pricing mechanism not only maximizes organizational benefits but also outperforms in all aspects of traditional performance measures compared to frequently used database scheduling techniques, such as first-come-first-served, earliest deadline first and least slack first.
(*User Preference*; *Information Services*; *Electronic Commerce*; *Response Time*; *Real-Time Databases*)

1. Introduction

In this paper we present a priority pricing mechanism to manage *negative externalities* in the operation of fixed-capacity real-time databases (RTDBs) that provide timely information services to users within organizations. The priority pricing mechanism lets users select a level of usage that maximizes the overall organizational benefit. We present the design and the implementation of this priority pricing approach, which acts as a natural admission control and a scheduling technique, as an online adaptive scheduling mechanism for RTDBs. We also present simulation results to demonstrate that our pricing mechanism not only

1047-7047/00/1102/0177$05.00
1526-5536 electronic ISSN

maximizes organizational benefits, but also outperforms frequently used scheduling techniques, such as first-come-first-served, earliest deadline first, and least slack first, with respect to traditional database performance measures (e.g., miss ratio, average tardiness). This research seamlessly integrates economic principles into RTDB query processing to provide timely information services.

Timeliness of information services is important in the present business environment with delay sensitive users. From an economic perspective delay sensitivity can be modeled as delay costs. When timeliness suffers, the collective delay costs can be substantial at the organizational level and can adversely affect organizational benefits from information services (Dewan and Mendelson 1990). In this context RTDBs, where the utility is measured by the responsiveness to user queries, play a central role in organizations by providing timely access to relevant information. In the RTDB environment, responsiveness is affected dramatically not only by the CPU speed but also by issues such as overload management, admission control, prioritization, and resource scheduling (e.g., CPU, disks and memory) (Abbott and Garcia 1992, Ramamritham 1993). However, commercial databases are not designed to support timeliness criterion because they provide no support for prioritization and dynamic resource allocation schemes (Jhingran 1996) and lack sophisticated admission control mechanisms.

Much of the research in RTDBs has been to address the issue of timeliness criterion (Bestavros 1996, Ramamritham 1993). However, the techniques suggested in the computer science literature to improve timeliness have provided little or no additional benefits, while largely ignoring how users value information. However, there is a rich literature in queuing theory (some within a microeconomics framework) addressing some of these issues in the organizational context (e.g., Dewan and Mendelson 1990, Mendelson 1985, Mendelson and Whang 1990, Westland 1992). Generally, timeliness is affected when individual users expand their usage of limited database processing capacity. This leads to negative (congestion) externalities; that is, each user's decision to utilize the database server imposes additional delay costs on the rest of the users and the organization. In an organizational setting, where there is operating transparency and high degree of trust, a pricing approach can manage and regulate user demand for computing resources. Several researchers have suggested the pricing approach for efficient allocation of computing resources (Mendelson 1985, Dewan and Mendelson 1990, Mendelson and Whang 1990, Stidham 1992, Westland 1992).

We apply prior work in the domain of pricing computing resources to RTDBs. We derive welfare maximizing priority prices and demonstrate the viability of such pricing for scheduling jobs for RTDB services. In our priority pricing approach, a premium is associated with higher priority jobs because these jobs impose additional (delay) costs on lower priority jobs. Additionally, we assume users submit jobs in various classes and are heterogeneous in how they value information and incur delay costs. Users submit requests to the database through agents that capture their cost and delay expectations (similar to MARIPOSA[1] in Stonebraker et al. 1994, 1996). The database matches user preferences against its own schedule of priority, price, and delay expectations for each job class (henceforth referred to as priority-price-delay schedule) and admits only those jobs with positive expected net benefits. The priority-price-delay schedule provides a natural admission control and discourages users who value information low from using database resources. This approach is significantly different from the traditional admission control approaches suggested in the computer science literature on RTDB. Typically, admission control is enforced either by a *multiprogramming* level that enforces the number of jobs that can be active in the system or through complex, rule-based, dynamic policies. Our research provides a mechanism to incorporate user preferences and organizational benefits into admission control strategies.

Earlier studies in priority pricing take an abstract view of internal pricing for the management and control of information processing services with an intention to provide normative insights. These analytical

[1]The prototype of the economic paradigm-based MARIPOSA, a wide-area distributed management system, is available at ⟨http://mariposa.cs.berkeley.edu⟩, the University of California, Berkeley.

models cannot be directly operationalized as an online resource allocation mechanism. There are several reasons:

1. Existing studies rely on steady-state queue lengths based on known demand patterns to decide whether or not a job should be submitted to the database. Furthermore, prices are meant to be static for long periods of time. These assumptions may not always perform well in the short run as temporary database overload could result in complete shutdown of the system (similar to "thrashing" in an operating system (Silberschatz et al. 1992)). This is because the effects of congestion in databases or real-time systems are known to be highly nonlinear (Stankovic 1988, Westland 1992).

2. Analytical models have to make restrictive assumptions to make the model tractable. They ignore the operational intricacies, such as data conflicts, memory size, input/output (I/O) requirements, and intermediate waiting times at CPU and disks. However, these operational intricacies impact database performance differently. For instance, the same query sent to two servers with different CPU speeds may retrieve information from the slower CPU server first if the required data (in terms of pages in computer science terminology) are already in main memory. It is also difficult to incorporate database performance metrics, such as jobs processed on time.

3. Many studies jointly optimize the arrival rates and service capacity that can be used for initial design. However, from an operational perspective, database capacity is fixed and the database system has to manage the service requests.

Given the reasons stated above, the theoretically derived prices need to be dynamically revised based on the changes in the demand structure and the observable attributes of the database. We provide an implementation strategy for this dynamic (incremental) adjustment of prices to regulate the flow to the database system. We evaluate the pricing approach and the implementation strategy through simulation. Simulation is a widely accepted methodology to evaluate database resource allocation techniques because it allows us to analyze the sensitivity to various constructs not included in the analytical model for the reasons of tractability discussed earlier (e.g., studies include Abbott and Garcia-Molina 1992, Agrawal et al. 1987, Stankovic 1988). We also use simulation to compare traditional approaches for database scheduling and admission control with the natural admission control and scheduling provided by the pricing policy. We compare our approach with widely used first-come-first-served, earliest deadline first, and least slack first techniques along two dimensions: Traditional database performance metrics, such as miss ratio, and economic objectives, such as organizational benefits. We bridge the gap that exists between analytical studies and simulation studies.

This paper is organized as follows: §2 discusses the relevant research in the areas of economics and computer science and argues the merit of incorporating economic principles into RTDB management. Section 3 discusses an economic approach to managing RTDB, an analytical model for pricing using economic theory, and a methodology for overload management and price recomputation. Sections 4 and 5 provide the simulation model and the results of our study. We conclude in §6 with a brief discussion of future research.

2. Related Work

The related work extends to two broad areas of research: Pricing in service facilities using queuing theory within a microeconomic framework, and the RTDBs in computer science. We first discuss related work in pricing.

Several researchers have studied the issue of queuing delays and the resulting negative externality within a microeconomic framework in the context of traditional service facilities, such as networks and computer systems. Naor (1969) first discussed regulating queue lengths by levying tolls. Kleinrock (1967) discussed priority pricing as a means to receive services earlier using a schedule of price and delay. Yechiali (1971, 1972) examined the optimal joining rule for GI/M/I and GI/M/s queues. He showed that the optimal policy to join a queue is when the queue length is below a threshold value and that the consumers' myopic decisions do not result in a socially optimal decision for any arbitrary price. Several other studies have extended and generalized Naor's model to understand the individual joining behavior (including balking and reneging) and socially optimal joining behavior in single queue systems. Mendelson (1985) and

Mendelson and Whang (1990) derived incentive compatible prices for M/M/1 queues that lead to socially optimal results when consumers make a myopic decision to join the queue based on prices and expected waiting times. Westland (1992) extended the work of Mendelson and Whang (1990) by examining the demand price elasticity and derived the profit-maximizing prices for a monopolistic setting. He argued that in a competitive setting, the prices would be set somewhere between the two extremes of monopolistic prices and social welfare prices. Dewan and Mendelson (1990) investigated optimal allocation decisions taking into consideration both users' delay cost and the capacity cost. Stidham (1992) extended this work further by looking at the long run-problem where capacity can be treated as a decision variable.

Stahl and Whinston (1994) derived socially optimal prices in a network setting with a generic queue waiting time structure. In this model, users choose an arrival rate myopically based on the prices and expected waiting times. Gupta et al. (1996, 1997) extended the work by Stahl and Whinston (1994) by deriving priority prices in a network setting and provided a computational mechanism for computing prices via an adaptive pricing mechanism. Other studies in the network setting that use an economic approach include Cocchi et. al (1993), Shenker (1995), Giridharan and Mendelson (1994), Mackie-Mason and Varian (1995), and Kurose and Simha (1989). Our analytical model is based on Gupta et al. (1997) and extends and applies it to the RTDB environment.[2] We assume that users make myopic decisions regarding their flow rate (or equivalently, the decision to join or balk from a queue) and present a welfare-maximization model to derive priority-prices.

A number of studies have investigated resource techniques in RTDBs from an engineering viewpoint with some success. Haritsa et al. (1991), Ramamritham (1993), and Yu et al. (1994) provide a good overview on RTDB resource allocation techniques. Principal among scheduling techniques are earliest deadline first (EDF), least slack first (LSF), shortest processing time first (SPTF), weighted priority (Huang et al. 1989), and various extensions of EDF. Some extensions of EDF are adaptive earliest deadline (AED) (Haritsa et al. 1991) that incorporates an admission control into EDF policy, and adaptive earliest virtual deadline (Pang et al. 1992), where AED was extended to make it fair to all job sizes. These resource-scheduling policies perform differently at different workloads. EDF is known to perform better in moderately loaded systems, while LSF performs better in overloaded systems (Abbott and Garcia-Molina 1992). Most of these studies neglect user values and delay costs and assume free database access. Furthermore, these studies assume job deadlines are known a priori. An exception to these studies is MARIPOSA (Stonebraker et al. 1994, 1996), a wide-area distributed database system, where the query processing is based on cost-delay curves. This system neglects future arrivals of higher valued queries and assumes that the cost is based on the load at the time of bidding. Our research is different from the previous studies in that we price jobs in a dynamic environment taking into account both current and expected future arrival rates. One of the key reasons for proposing an economic paradigm-based architecture in MARIPOSA is that cost-based optimizers do not scale well and do not respond well to site-specific-type extension, access constraints, charging algorithms, and time-of-day constraints.

Konana et al. (1996a,b) note that economic issues impact several database issues such as data classification, transaction scheduling, buffer management, admission control, concurrency control,[3] data replication, and storage management. Timely execution of user jobs may not be an issue when the database server has less workload. In fact, most scheduling disciplines suggested for database transaction processing perform equally well at low workloads (Ramamritham 1993, Abbott and Garcia 1992, Bestavros 1996, Konana 1995). However, as the server becomes congested (referred to as "overload" conditions in the computer science literature), the resource allocation mechanism can have a significant impact on the performance.

[2]Note that for a M/M/1 queue our model reduces to the model of Mendelson and Whang (1990).

[3]Concurrency control is needed to ensure that concurrent transactions do not interfere with each other's operation.

3. Priority-Based Pricing Model

In this section, we develop a social welfare model for pricing RTDB services within organizations. As Mendelson and Whang (1990) and Westland (1992) note, social welfare prices may not be able to recover the costs if congestion-based externality prices are the sole revenue sources. However, using social welfare prices has the advantage of maximizing organizational benefits from computing resources. Prices in these environments play a role of transfer prices rather than revenue source. Westland (1992) discusses the role of congestion-based pricing for cost recovery within a two-part tariff scheme, where there is a fixed charge for access (subscription charge) and externality-based prices. However, we focus on resource allocation and the resulting operational characteristics of the database and do not focus on cost recovery or the profit maximization problem in this paper. We focus on externality pricing to influence users' decisions regarding the level of their usage to optimize organizational benefit. Subsequently, we develop a strategy for real-time implementation of derived prices and investigate the performance of the database.

We first discuss an abstract framework of the database job processing and pricing scheme, and then develop an analytical model to derive the optimal prices. In our framework, the users submit queries using client (software) agents. These agents capture and transmit user preferences in terms of cost and delay to the information server. The information server will then find the best match from the precomputed priority-price-delay schedule. Once a job is accepted in a particular priority class, it remains in the same priority until completion. We assume that user requests have a *soft* deadline (Ramamritham 1993), that is, jobs will continue to execute until completion even when their deadline expires (note that explicit job deadline is not used in the analytical model, but is used in the simulation for performancee valuation). We relax this assumption and analyze through simulation the effect of jobs dropping out of the queue when their deadlines are reached. The pricing mechanism implicitly considers processing requirements, expected intermediate waiting times, user values, and delay costs. For the implementation of a pricing mechanism, the knowledge of consumers' private value for information services is not required. However, for expository purposes we use these values in the model description. The actual price computation does require information on consumers' delay cost, which can be estimated from the consumer choices as discussed in Gupta et al. (1997) (see Appendix A).

3.1. RTDB Model

The RTDB is a disk-resident shared memory multiprocessor system. The server is associated with a priority queue at the CPU and the disks. A job may have to wait at the CPU and disk queues at multiple instances due to disk access and time-sharing.[4] In this paper, we consider read-only queries and ignore updates. Hence, any unpredictability in processing time due to concurrency control is eliminated (Ramamritham 1993). Thus, we assume that the database processing capacity is a function of the CPU processing capacity measured in terms of pages processed per unit time. Table 1 shows the notation and objectives functions used in the model and subsequent simulation experiments.

Each request arriving to the database can be preanalyzed off-line to identify the size. We measure the size of a job, s, in terms of the number of pages to be fetched. Let s_j be the size of the job in a known job class j (j 1,2, . . , J).[5] Every job is assigned a particular priority class k (k 1,2, . . . , K) where k 1 is the highest priority class.

During processing of a job, the required data (in terms of pages) are retrieved from the disk into main memory. Each page requires q time units to fetch from the disk and p time units to process at the CPU. Between processing of pages, a job may have to wait in queues to access the disk and the CPU (intermediate queues are not modeled for analytical tractability; however they are modeled in the simulation). Let ω_{jk} be the expected total waiting time at CPU and disk queues for each job class j in priority class k. Then the total expected time, τ (j,k), to process a request in job

[4]While the number of CPUs or disks is not used in the analytical model, they are required for simulating the real environment discussed in §4.2.

[5]Transaction size has a direct impact on the processing or response time. The higher the number of pages/data items to be fetched, the higher may be the number of input-output (I/O) requirements.

Table 1 Notation and Objectives Used in the Model and Simulation

Type	Notation and objectives	Description
	i	Index to represent a user (i 1, 2, . . . I)
	j	Index to represent a job class (j 1, 2, . . . J)
	k, h	Index to represent a priority (k, h 1, 2, . . . K)
Workload parameters	λ	Arrival rate of jobs
	ω_{jk}	Waiting time in priority k for job class j.
	$\tau(j,k)$	Expected time to process a request in job class j and priority k. (Note: t_{ij} is used to indicate actual process time in §4)
	p	CPU time to process per page
	q	Disk access time per page
	s	Number of pages to be processed in a job
	v	Database capacity in terms of pages.
User related parameters	V_{ij}	Instantaneous value of information for user i for job class j.
	u_i	Overall net benefit to user i.
	δ_{ij}	Delay cost per unit time for user i in job class j.
	c_{ij}^*	Minimum cost of accessing information in job class j for user i.
	d_{ij}	Deadline of a job submitted by user i in job class j.
	r_{jk}	Estimated access price at priority k and job class j. (t) r_{jk} is used to represent estimated access price at time t.
	$\hat{r}_{jk}(t)$	Actual implemented price at time t.
	α	Adjustment parameter for access price ($\alpha \in (0,1)$)
Function	$\Omega_k(\bullet;v)$	Waiting time function for a given database capacity v.
Matrix	Λ	Matrix of arrival rates for each priority k and job class j.
Objective Function	W	Systemwide net benefits (During simulation we use benefits per unit time)
	S	Consumer surplus (During simulation we use surplus per unit time)
Other Simulation Objectives[a]	Miss Ratio	Percentage of jobs missing deadline.
	NMR	Miss ratio normalized by job size.
	Average tardiness	Average lateness of jobs completed after the job deadline

[a]These simulation objectives are traditional database-related performance measures required for analysis.

class j and priority k is the sum of the expected waiting time, ω_{jk}, and the total time to process pages at the CPU and disks.

3.2. Users and Demand Function

We consider the RTDB system as an economic system that serves each user i (i 1,2, . . I). A user i may be a group of users, such as marketing, finance, or customer service group. We assume service needs for user i as a stochastic arrival process with a specific arrival rate for a job class j. Each user submits a job with a cost and delay expectation to the database. The database then assigns an appropriate priority k to each request. Let λ_{jk} denote the average flow rate for user i for job class j in priority k. Below we describe the value function, delay costs, and the decision structure for user requests.

Value Function. We assume that a user i has an instantaneous value (i.e., at delay time zero) $V_{ij}(\lambda_{ij})$ for a request in job class j with a realized[6] flow rate λ_{ij} where $V_{ij}(\lambda_{ij})$ is continuously differentiable, nondecreasing, and concave. We assume that the user valuation of information depends only on his average flow

[6]We define realized flow rate as the flow rate resulting from actual submission of requests. Note that some of users' requests may not be submitted, either because the required performance criterion cannot be met, or the cost to the user may be too high.

rates and is independent of other arrival and service processes. In an intranet setting, for example, a customer service group has a very high value for customer information since it is required to provide quick service. However, the net value to a user is less than $V_{ij}(\lambda_{ij})$ because the value of the information diminishes with elapsed time (delay cost) and there is a cost to retrieve this information. In the customer service example, delays in retrieving customer information lead to customer dissatisfaction and, therefore, the organization's net benefits decreases.

Delay Costs. We assume a linear delay cost for all users for analytical tractability.[7,8] Each user i experiences a delay cost, δ_{ij}, per unit time for a given job class j. Therefore, the expected delay cost for job class j of priority k is $\delta_{ij}\tau(j,k)$. The estimation of delay cost appears to be a difficult task because users have no incentive to truthfully report their true delay costs. However, one can approximate user delay costs by monitoring user past choices with respect to priority, delay expectations and cost for each job class. A brief description of the methodology is discussed in the Appendix.

Decision Structure. We assume that each user maximizes his net benefits in the presence of access price and delay costs. Let the cost to access information from the database for job class j and priority k be r_{jk}. Then the minimum cost of accessing information in job class j from the database with a given waiting time and price is:

$$c^*_{ij}(r,\omega) = \min_k [\delta_{ij}\tau(j,k) + r_{jk}]. \quad (1)$$

That is, the optimal cost, c^*_{jk}, is the minimum of the sum of delay cost ($\delta_{ij}\tau(j,k)$) and access cost (r_{jk}) over all priorities. Equation (1) determines the appropriate priority class, k, that delivers the information at the lowest possible cost. Note that the delay cost term $\delta_{ij}\tau(j,k)$ and access cost r_{jk} are inversely proportional. This is because higher priority is associated with higher access price but with lower delay costs, while lower priority is associated with lower access price but with higher delay costs. However, the delay costs depend on the waiting times, which are themselves a function of the realized flow rates into the system; that is, the waiting times depend on the current demand and expected future demands from all users.

To arrive at the realized flow rates into the system for each user, we need to consider the overall net benefit to the user. The overall net benefit, u_i, of user i for a given realized flow rate λ_i, rental cost, and waiting time can be defined as:

$$u_i(\lambda_i,r,\varphi) = \sum_j (V_{ij}(\lambda_{ij}) - \sum_k \lambda_{ijk}c^*_{ij}(r,\omega)). \quad (2)$$

Equation (2) states that the overall net benefit to each user i is the instantaneous value of jobs minus the cost of delays and access price summed over all jobs in each job class. If we assume that each user i will choose arrival rates that will maximize his net benefits then the realized flow rates from each user are determined by the following:

$$\begin{aligned} &\partial V_{ij}(\lambda_{ij})/\tau\lambda_{ijk} \geq c^*_{ij}(r,\omega) \ \forall\ j \text{ and } k, \\ &\text{and } \partial V_{ij}(\lambda_{ij})/\partial\lambda_{ijk} < c^*{}_{ij}(r,\omega) \Rightarrow \lambda_{ijk}(r,\omega) = 0. \end{aligned} \quad (3)$$

That is, a user will evaluate his marginal benefits before his request is submitted for processing. If the marginal benefit is greater than or equal to the access cost, then the request is submitted, otherwise he balks. Then, the realized flow rate from each user is the result of all submission decisions. This implies that the information server can potentially manipulate the flow into the system to achieve a desired response time for all jobs.

3.3. Optimal Resource Allocation

In this subsection, we develop the methodology for generating the priority-price-delay schedule that maximizes the net benefits given the demand function,

[7]Different types of delay costs have been used in RTDB (Abbott and Garcia-Molina 1988) and queuing theory related (Dewan and Mendelson 1990) studies. The deadline delay cost structure in Dewan and Mendelson (1990) refers to soft deadlines in RTDB literature. In a hard deadline, a significant negative value is imposed on the system when a transaction is not completed on time. In a firm deadline, a transaction has no value once the deadline is crossed.

[8]Note that delay cost is an implicit factor; rarely will users know their delay cost factors. Instead users specify deadlines and the delay costs have to be determined from it. In the database literature, often piecewise linear delay costs are considered, where after the deadline is reached there is no further decay in value, and the net user value for service remains zero. Assuming continuous linear delay costs, on the other hand, makes benefits negative once the deadline is reached.

user's value function and delay costs, and decision structure. The database is said to be in equilibrium when the demand for services, $\Sigma_{i,j,k}\ \lambda_{ijk}s_j$, equals the processing capacity ν (in terms of pages) of the database. However, in any stochastic process, if the arrival rates are equal to the capacity then the waiting times $\to \infty$. Any feasible pricing mechanism has to provide a rationing mechanism that ensures that the demand is less than the capacity at all times. Let Λ represent the matrix of job arrival rates, λ_{jk}, at all priority and job classes. For a given database capacity ν, we can represent the waiting times ω_{jk} as a function of the arrival rate matrix Λ.

$$\omega_{jk} = \Omega_k(\Lambda;\nu)\ \forall\ j, \tag{4}$$

where $\Omega_k(\Lambda;\nu)$ is assumed to be continuously differentiable with respect to λ_{jk}, strictly increasing and convex as long as demand for services is less than the database capacity and $\Omega_k(0;\nu) = 0$. Differentiability is implicitly assumed in related work to obtain optimality conditions. Furthermore, $\Omega_k(\Lambda;\nu) \to \infty$ as demand approaches the database capacity. We assume that $\partial\Omega_h/\partial\lambda_{jk} \geq \partial\Omega_h/\partial\lambda_{jk'}$ for all $k < k' \leq h$ ((note that the highest priority is 1 and the lowest priority is K); that is, the incremental waiting time imposed on priority h jobs is greatest for jobs arriving with the highest priority (*negative externality*). The waiting time for any request in the higher priority is notaffected by the new requests in lower priority, while any new request in the higher priority affects all lower priority requests.

To derive the optimal trade-off we need to define a systemwide welfare function. We maximize user benefits, that is, benefits of all users minus the delay cost for all users:[9]

$$W(\lambda,\omega) \equiv \sum_{i,j} (V_{ij}(\lambda_{ij}) - \delta_{ij} \sum_{k} \lambda_{ijk}\tau(j,k)). \tag{5}$$

We now seek an allocation of demands, λ_{ijk}, and waiting times ω_{jk} that maximizes $W(\lambda,\omega)$ subject to Equation (4). We solve the global maximization problem using the standard Lagrangian method and Kuhn-Tucker (K-T) conditions for optimality (Bazaraa and Shetty 1993). Substituting the Lagrangian multiplier to the K-T conditions, we find that the allocation of demands, λ_{ijk}, will satisfy the K-T conditions when:

$$r_{jk} = \sum_{h} [\partial\Omega_h/\partial\lambda_{jk}] \sum_{i,j} \cdot_{ij}\lambda_{ijk}. \tag{6}$$

The rental price is the welfare-maximizing price for database access in priority k and is equal to the average cost of aggregate delays ($\Sigma_{i,j}\ \delta_{ij}\lambda_{ijk}$), weighted by the waiting-time/throughput trade-off ($\partial\Omega_h/\partial\lambda_{jk}$).

Given our assumptions about the waiting time function, Ω_k ($\bullet;\nu$), the rental prices are highest for highest priority class (priority 1) and decreases as the priority class decreases, i.e., $r_{jk} > r_{j(k+1)}$. Hence, one could think of r_{jK}, where K is the lowest priority, as the base price, and ($r_{jk} - r_{jK}$) as the premium for accessing the higher priority k for job class j.

Note that Equation (6) is not an explicit formula for rental price r_{jk}, since r_{jk} enters the right-hand side through λ_{ijk} and the resulting arrival rate matrix Λ.[10] Instead of using traditional fixed-point methods of computing prices we favor an approach which is motivated by the classical tatonnement process (Hahn 1982). Our approach has the benefit of providing an adaptive mechanism to compute and implement prices in real-time while not requiring the knowledge of the delay factors.

In summary, we generate an optimal priority-price-delay schedule that maximizes systemwide welfare. In this schedule every job class j is associated with a priority, a price, and a delay. At optimality, the rental cost for a particular job class and priority class is equal to the aggregate delay cost of users wishing to use the system for a given time period. When a user's request arrives with service characteristics and a price range, the server agent can identify the best priority for that job by minimizing the expected total cost (delay cost + price). This priority is then used for resource allocation (CPU and Disk) in the RTDB. Jobs in the same priority class are prioritized on a first-come-first-served (FCFS) basis.[11] A request may not be serviced if the price a user is prepared to pay is infeasible with

[9]Note that the price is the transfer of revenue from the users to the provider. Therefore, the sum remains in the economic system and, hence, is part of the total welfare.

[10]This is true for all microeconomic models where demand is price elastic.

[11]Note that our model assumes a fixed number of priority classes and then uses FCFS for scheduling.

the service requirements. In the next section, we describe how the priority-price-delay schedule can be operationalized in a database environment to manage overload conditions.

3.4. Operational Model—Overload Management and Price Recomputation

Our model estimates the expected delay times based on the predictions of the arrival rates for various job classes and priorities. However, the variability in the actual arrival rates may result in overload situations where the system consistently processes user requests late. While stochastic equilibrium results are valid when the system is run on a long-term basis, it may not be applicable for a short-term increase in arrival processes as temporary variability may dramatically alter database performance. Overload situations may lead to performance degradation fairly rapidly, sometimes leading to system downtime, such that none of the jobs will be processed, an effect similar to "thrashing" observed in operating systems (Silbershatz et al. 1992, Stankovic 1988). Therefore, when the system observes overload conditions some actions must be taken to control admission at the earliest. In our case, overload implies that the priority-price-delay schedule must be adjusted to reflect the actual demand for system resources, which in turn changes the admission control. That is, when the demand increases, the prices are recomputed to induce further restrictions on the admission. On the contrary, if all the jobs are processed well within the delay estimates, then the price schedule must be revised downwards to allow more jobs. This priority-price-delay schedule is adjusted incrementally as discussed below.

Our price computation approach measures the average flows to the database and predicts the expected demand and waiting times. We compute the priority-price-delay schedule at discrete times where the time interval between any two successive times is some constant, t_c. Let $\lambda_{jk}(t)$ denote the current time-averaged estimate of λ_{jk}, let $\omega_{jk}(t)$ denote the current time-averaged waiting time for a given job class and priority at time t, and let $r_{jk}(t)$ denote the estimated prices from the analytical model at time t. Let the actual (implemented) prices at time t be $\hat{r}_{jk}(t)$. We compute rental price $r_{jk}(t + t_c)$ at time $(t + t_c)$ by using the estimates of $\lambda_{jk}(t)$ and $\omega_{jk}(t)$ at time period t. Because $r_{jk}(t + t_c)$ is based on short-term observations of arrival rates and waiting times, it may be quite volatile in a stochastic environment. To reduce the chances of instabilities resulting from overresponsiveness, $r_{jk}(t + t_c)$ is used as an indicator of whether to increase or decrease the previous price $\hat{r}_{jk}$. The prices for period $(t + t_c)$, $r_{jk}(t + t_c)$, is set to

$$\hat{r}_{jk}(t + t_c) = \alpha r_{jk}(t + t_c) + (1 - \alpha)\hat{r}_{jk}(t) \qquad (7)$$

where $\alpha \in (0,1)$ is the adjustment parameter determined empirically. The price differences between two time periods are larger when the value of α approaches one.

In the next section, we provide details of the simulation study and report results regarding the system-performance with pricing mechanism and contrast the results with those obtained from several commonly used database scheduling and admission control mechanisms. We first motivate the simulation and thenprovide the simulation model that uses the parameters described in our analytical model.

4. Performance Evaluation

4.1. Analytical Versus Simulation Analysis

The analytical models for evaluating computer systems using queuing theory typically make stochastic assumptions and involve problem abstractions. As discussed earlier, the operational intricacies are generally neglected for analytical tractability (Stankovic 1988). Such assumptions and problem abstractions are valid, as the purpose of these studies is to provide normative insights (behavior of the system). Similarly, the analytical model in this paper generates the priority-price-delay schedule based on problem abstraction, neglecting the complex operations of the database systems. However, our model is proposed to be an *integral part* of the RTDB by adapting dynamically to the state of the database system and to changes in the demand structure (generally referred to as the online adaptive scheduling mechanism in computer science literature). Since we intend for our model to be a basis for an online resource scheduling technique, the performance must be evaluated within a database environment where some of the modeling assumptions are relaxed.

We use a simulation approach to evaluate the RTDB performance-related issues. The specific objectives of the simulation study are:

1. Evaluation of the performance of our model under short-term increase in arrival processes. Analytical models use convenient stochastic assumptions that are justified by large populations and stable operating conditions (Stankovic 1988). However, database systems (particularly real-time databases) can observe highly nonlinear performance degradation under short-term overload situations leading to complete shutdown of the system. In such situations, we need to evaluate whether our priority-price-delay schedule recomputation (§3.4) will return the system to equilibrium fairly rapidly. If the priority-price-delay schedule computed based on short-term observation using Equation (7) diverges significantly from the optimal schedule, then our approach may not yield expected economic gains. The simulation also allows us to observe the types of jobs that are allowed into the system during short-term increase in arrival processes.

2. Testing the performance of the RTDB system using our model with respect to traditional performance measures, such as the number of jobs processed on time (or the number of jobs processed late), throughput and tardiness. These performance measures, typically used in computer science literature, are difficult to incorporate into microeconomic models. While our model is designed to maximize organizational benefits, it is interesting to observe whether it dominates the performance with respect to traditional measures as compared to first come first served (FCFS), earliest deadline first (EDF), and Least Slack First (LSF) (Abbott and Garcia-Molina 1992, Haritsa et al. 1991).

3. Evaluation of the robustness and the equilibrium results when certain model assumptions are dropped. In our model, we assumed that once accepted a job is executed until completion even when the performance requirements (e.g., timeliness requirements) cannot be satisfied. We test the system performance and evaluate economic benefits by relaxing this assumption.[12] We drop user requests from the queue during processing if the actual waiting times are higher than the expected waitingtimes.

[12]We are indebted to the anonymous referee that suggested we test this assumption.

4.2. Simulation Model

We implemented a computer simulation model that captures the main elements of a RTDB system in CSIM, a discrete event simulation language (Schwetman 1991). The simulation uses a queuing model ofa single-site, shared-memory, disk-resident database system. This model is a modified version of the simulation model provided in (Abbott and Garcia-Molina 1992, Agrawal et al. 1987, Konana 1995).

The simulator consists of six active modules (namely, *request generator, request preanalyzer, transaction manager, resource manager, statistics collector,* and *price generator*), and one passive component, namely, the *database*. The *price generator* implements our priority-pricing mechanism. We neglect concurrency control since we are dealing with read-only jobs. Figures 1 and 2 show the logical and physical simulation models. The *database* is modeled as a set of pages distributed uniformly across all disks. Therefore, a given page can be mapped to a specific disk. As discussed in §3.1, there is one queue for all CPUs while each disk has its own queue.

The *request generator* generates user requests from a Poisson process with a mean arrival rate that is set at various values during simulation. This arrival rate captures the exogenous arrival of request for database services. Each request generated is assigned a particular job class j. The size (in terms of pages) of a request, s_j, is chosen uniformly from a range. Every incoming request is associated with an instantaneous value V_{ij} and a delay cost factor δ_{ij} drawn from normal distributions with mean and standard deviations (μ_v, σ_v) and (μ_d, σ_d), respectively. Given the V_{ij} and δ_{ij}, we compute an expected deadline[13] (d_{ij}) for a request as follows:

$$d_{ij} \quad V_{ij}/\delta_{ij} \tag{8}$$

The *preanalyzer* matches user requests according to Equations (1) and (2) in §3.2 with the priority-price-delay schedule generated by the *price generator*. The *price generator* uses Equations (6) and (7) to set prices $r_{jk}(t)$ at time t. The *preanalyzer* acts as an admission controller by accepting requests for which the net benefits are positive.

[13]We need to assign a deadline to each request to monitor whether or not requests are executed within their deadlines and to compute consumer surplus.

Figure 1 Simulation Model

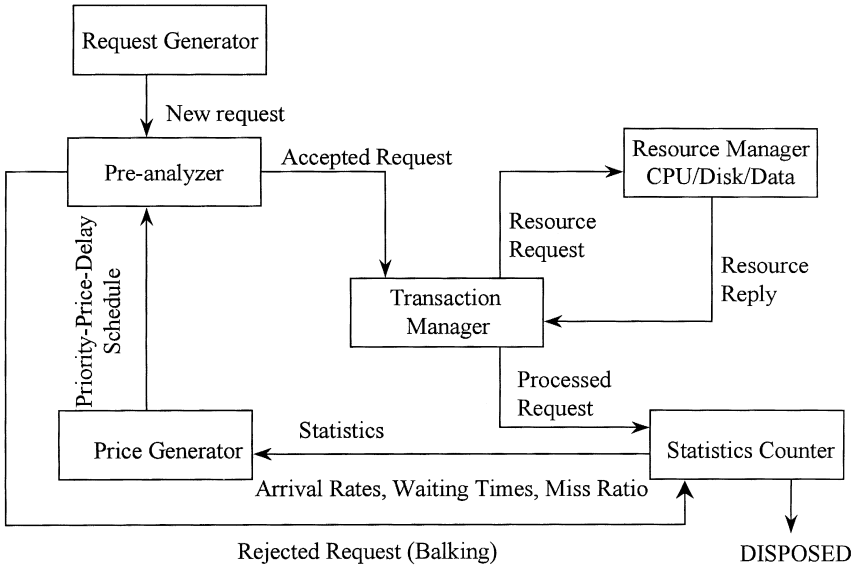
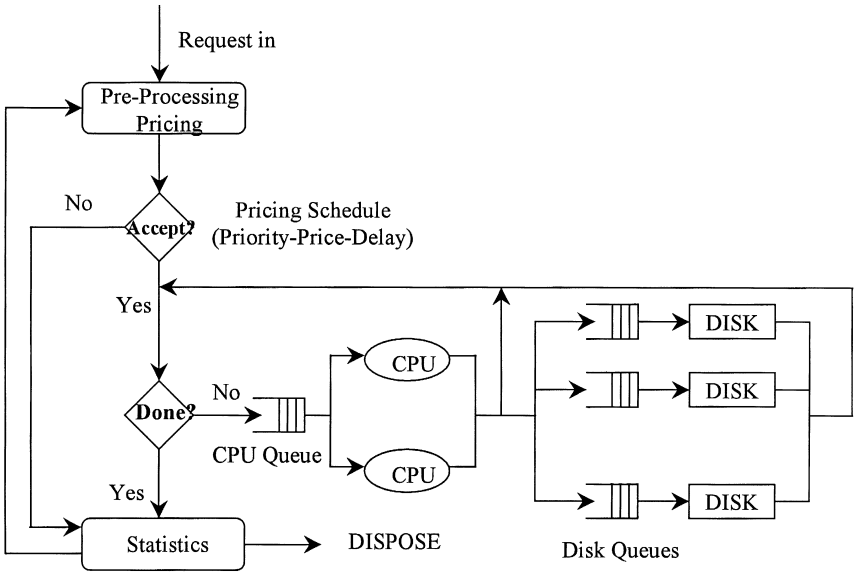


Figure 2 Physical Queuing Model



Once a request is accepted, it is assigned an appropriate priority and is executed until completion. The *transaction manager* is primarily responsible for implementing the scheduling policy. The *transaction manager* closely interacts with the *resource manager* and monitors the completion of a request. When a request is completed, the request is removed from the system and sent to the *statistics collector*. We do not explicitly implement a full memory management system. Most performance evaluation literature assumes a probabilistic database buffer pool rather than modeling individual pages as it is likely to affect all algorithms

equally (Abbott and Garcia-Molina, 1992). If the page is not in memory, then an input-output service request is sent to the appropriate disk.

The *price generator* computes the prices based on Equation (6) while the price recomputation is based on Equation (7). The parameter estimates are derived from the *statistics collector*. In the simulation we update the price and waiting time information at fixed intervals of time.

4.3. Resource and Model Parameters

Table 2 provides resource parameters. These parameters have been adopted from published simulation literature on real-time databases (Abbott and Garcia-Molina 1992, Konana 1995, Kim and Srivatsava 1991, Haritsa et al. 1991) and modified for our study.

We assume one CPU and two disks. The number of disks is assumed to be twice the number of CPU to make balanced resource utilization, as opposed to being either strongly CPU bound or strongly I/O bound (Agrawal et al. 1987). The database is assumed to consist of 1,000 pages. These assumptions are made in numerous other simulation studies in the computer science literature (e.g., Abbott and Garcia-Molina 1992, Haritsa et al. 1991). Other parameters, such as p (i.e., CPU time to process a page) and q (i.e., disk access time for each page) are also borrowed from the above simulation studies (These parameters are used in §3.1 in describing the RTDB model.) We assume that each request has a maximum of 20 pages (size s) and is drawn from 40 job classes. Since no real data are available for instantaneous user values (V_{ij}) and delay costs (δ_{ij}), we have assumed distributions based on the law of large numbers (i.e., normal distributions)[14] (Gupta et al. 1997). We assume a mean and a standard deviation of (25,7) and (4,1) for user values and delay costs, respectively. We believe that, given the processing speeds and the data requirements in this model, we have chosen a conservative set of values. Thus, our benefit results are, at best, understated. Note that to get λ_i we specify an overall exogenous arrival rate λ. Since the database does not require the knowledge of who submitted a particular job for price computation (see Equation (6)) and we are only interested in systemwide benefits, we do not explicitly model the jobs for individual users i. We assume α 0.1 for price recomputation using Equation (7).

[14]In future studies, we intend to perform extensive studies for the robustness to these values similar to the study in Gupta et al. (1997b).

Table 2 Resource Parameter Values

Parameter	Meaning	Value
p	CPU time for each data page	10 msec
q	Disk access time for each data page	20 msec
J	Number of job classes	40
K	Number of priority classes	1

4.4. Performance Metrics

We evaluate the performance of our pricing model against traditional scheduling policies, such as *FCFS*, *EDF* and *LSF* using both the RTDB and economics related performance metrics. In the simulation, we compute net system benefits (W) according to Equation (9) (equivalent to Equation (5)) and consumer surplus (S), a key indicator of consumer welfare, according to Equation (10). Both of these statistics are collected on a per time unit basis for consistent presentation of results.

$$W = \sum_{i,j} \frac{V_{ij} - \delta_{ij} t_{ij}}{t_{ij}}, \tag{9}$$

$$S = \sum_{i,j} \frac{(d_{ij} - t_{ij})\delta_{ij}}{t_{ij}}, \tag{10}$$

where t_{ij} is the actual time delay faced by the user i for a job j. Note that t_{ij} is different from t_{ij} which is the expected time of completion and is used only for estimation of expected cost according to Equation (1). The deadline as computed by Equation (8) is d_{ij}.

To compare the database performance under pricing with other admission control and RTDB scheduling approaches, we use the traditional performance metrics from the RTDB literature, such as the *normalized miss ratio* (*NMR*) (Pang et al. 1992) and the *average tardiness*. The average tardiness is defined as the average lateness of jobs completed after the deadline. The NMR captures the fraction of the offered load that is not completed on time, weighted by job size. It is weighted by job size because some scheduling algorithms, such as *EDF*, seem to favor smaller size job classes (Pang et al.

1992). To minimize this bias, the *miss ratio* is normalized by the size. The *NMR* is computed as follows:

$$NMR = \sum_j \frac{s_j \times \textit{miss ratio}(s_j)}{s_j}. \quad (11)$$

Where *miss ratio* (s_j) denotes the miss ratio of jobs of size s_j and denotes the range of job sizes in the workload. The *miss ratio* is computed as follows:

$$\textit{miss ratio} = \frac{\#\ \textit{of jobs missing deadline}}{\#\ \textit{of jobs arriving}}. \quad (12)$$

5. Simulation Results

The simulation was run on HP workstations. Our study, with a wide range of parameter values, gives us confidence that the results presented here are robust against parametric disturbances. We are particularly interested in the system performance under overload conditions and comparing that performance against traditional approaches, such as *FCFS, EDF,* and *LSF.* Therefore, we opted to have a single processor in order to reach overload situations quickly.

The sets of experiments considered are shown in Table 3. We carried out two sets of experiments to analyze how the system behaves and to compare performance against traditional admission control and scheduling techniques. In the first set of experiments, we compare access pricing (Strategy 1a) against no pricing with admission control (Strategy 1b), and no pricing and no admission control (Strategy 1c). In each case, the scheduling discipline is *FCFS.* For Strategy 1a, the pricing itself acts as an admission control and, therefore, there is no explicit admission control. In Strategy 1c, every request that arrives to the database is allowed to enter the system. In Strategy 1b, requests that have positive expected net benefits are allowed to enter the system by disregarding the cost of accessing the databases. Therefore, more requests are allowed into the system in Strategy 1b compared to Strategy 1a, but significantly fewer than those in Strategy 1c. Although it may be obvious that the pricing scheme provides higher benefits, this experiment is carried out to observe the true behavior of the system. Apart from investigating the net benefits, *NMR* and tardiness, we are interested in observing the types of requests allowed into the system as the system becomes overloaded.

In the second set of experiments, we investigate how our pricing model performs against *EDF* and *LSF* scheduling techniques at various multiprogramming levels. To make a fair comparison across all the models, we first evaluate the *NMR*s of Strategies 2b and 2c under various multiprogramming levels with that of Strategy 2a. We then compare the benefits across all three techniques where *NMRs* are approximately the same. We conduct Experiment 2 at arrival rates 20 requests/second and 50 requests/second.

5.1. Experiment 1

In these experiments, we varied the arrival rate from 10 jobs/sec to 50 jobs/sec in increments of 10. Statistics were gathered based on the replication-deletion approach (Law and Kelton 1991). Each experiment consisted of 10 runs with a transient period of 1,000 time units and length of 10,000 time units each. Statistics collected were averages of these 10 runs.

5.1.1. Effect on Net Benefits. Figure 3 shows the net system benefits, *W,* (objective function of the analytical model) with and without the pricing mechanism. In all the cases, a user with a positive expected net benefit is admitted to the system. At low λ, pricing is not an issue because the system in not overloaded and every request's response time is satisfied. However, at higher λ the collective benefits with pricing (that is, with database access cost) is significantly higher than that without pricing (that is, without database access cost) at a 99% confidence level. The experiment was also repeated for a system with no admission control and pricing. In fact, as expected, the system provided significant negative benefits (the results of no admission control and pricing are not shown). The reason for higher system benefits with a pricing mechanism is that at higher arrival rates only those requests with higher values were admitted. This value-based admission control effectively blocks the requests with lesser values and those that require significant resources.

In our theoretical (and thus simulation) model, users decide a priori whether or not they will enter the queue based on the prices and the expected waiting times.

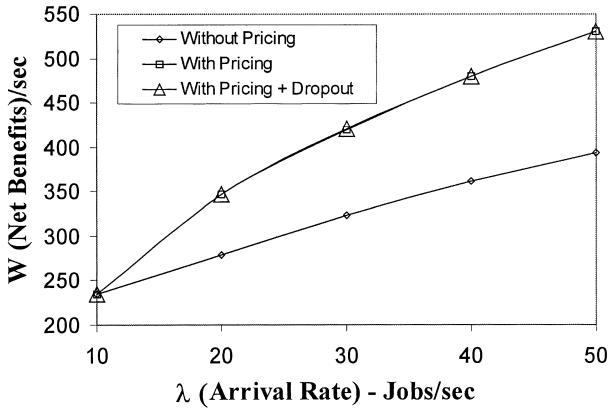
Table 3 Experimental Setup

	Strategy	Pricing	Admission Control	Scheduling Discipline
Experiment 1	1a	Yes	No* $(V_{ij} - (\delta_{ij} \times t_{ij}) - r_{jk}) \geq 0$	FCFS
	1b	No	Yes $(V_{ij} - (\delta_{ij} \times t_{ij})) \geq 0$	FCFS
	1c	No	No	FCFS
Experiment 2	2a	Yes	No $(V_{ij} - (\delta_{ij}\, t_{ij})) \geq 0$	FCFS
	2b	No	Yes Multiprogramming level of [20–70] requests	EDF
	2c	No	Yes Multiprogramming level of [20–70] requests	LSF

*Pricing acts as an admission control.

Since expected waiting times, by definition, mean that actual waiting time observed by a user could be higher (or lower) than the expected value, one might wonder about the behavior of marginal users who may experience longer than the expected waiting times. When this occurs, users with expected net positive benefits actually incur negative net benefits. Therefore, we experimented with a system in which users can monitor their net utility while they are in the queue and decide to drop out of the queue when the actual waiting time is too long (this drop out is not modeled in our analytical model). Figure 3 shows the net system benefits, W, when such a system is implemented. The results may seem surprising at first glance, as there is virtually no difference in net benefits when these marginal users drop out. However, this result can be explained when we analyze the equilibrium behavior. Consider what happens to the system benefits (i.e., aggregate user values minus aggregate delay costs). When a user request drops out, its value is not added to the system benefit, while the cost suffered by this request is added to the total delay cost. The system receives a negative impact from the request that drops out. However, when a request drops out, all the other requests in the queue benefit because their actual waiting times are now reduced by the service time that the dropped out request would have consumed. In equilibrium, the cost resulting from dropping a request and the benefit as a result of the reduction in delay costs of all the other users should balance out; our simulation results indicate that this is indeed the case. In the next section, we will also explore the effect of dropping marginal users on the database performance metrics.

Figure 3 Net System Benefits With and Without Pricing at Various Job Arrival Rates



5.1.2. Effect on Miss Ratio and Average Tardiness. In this experiment, we are interested in analyzing how the pricing mechanism performed when using traditional performance metrics. Figure 4 shows the miss ratio with pricing, pricing with drop outs, and without pricing mechanisms. At higher arrival rates, the miss ratio for the system without pricing was significantly higher than that with pricing. Surprisingly, the miss ratio in both cases decreases at higher arrival

rates. This became apparent on analyzing the actual miss ratios at various job sizes. In fact, at very high arrival rates, a sufficient number of smaller jobs that provided significantly more benefits than the larger-sized requests were admitted. The larger jobs consume more resources, while adding little to the collective benefits. At arrival rates between 10 and 20 requests per second, some large-sized requests were allowed to execute that affected execution of smaller sized requests, resulting in higher miss ratios. It is also important to note that the miss ratio does not dramatically increase when overload occurs, as observed in numerous studies in the computer science literature (Abbott and Garcia-Molina 1992, Haritsa et al. 1994). This behavior results from the pricing mechanism acting as a natural admission control and overload management mechanism.

In the case of pricing with dropout, the miss ratios were found to be higher relative to that of pricing and no dropout.[15] This has to be expected because, as explained earlier, the net result of drop out is that the system waiting time is reduced (fractionally). Thus, more marginal jobs may provide positive expected benefits. However, since these jobs are marginal and have smaller positive values, a larger number of them drop out, resulting in higher miss ratios.

The above results are consistent with the average tardiness of late jobs shown in Figure 5.[16] The average tardiness is actually reduced at higher arrival rates since larger jobs were blocked and smaller jobs were executed. Note that this behavior is analogous to the well known c-μ[17] scheduling rule in queuing theory in which between two jobs that cost the same, the smaller job is executed first. We also conducted experiments without either pricing or admission control. The miss ratio hits over 90% even with an arrival rate of 20 jobs/sec and, hence, is not shown in the figure.

5.2. Experiment 2

In this experiment we evaluated the performance of our pricing approach against *EDF* and *LSF* scheduling policies under various multiprogramming levels. The multiprogramming level implies that the maximum number of requests executing (active) in the system is fixed for each run. For example, a multiprogramming level of 50 implies that only 50 requests can be processed at any given time. We chose to conduct this experiment at arrival rates of 20 jobs/second and 50 jobs/second to simulate a moderately and a highly overloaded system, respectively. When the system is full any new request is rejected. In traditional databases, such requests are queued and allowed into the system as and when active requests are completed. Our modified multiprogramming approach provides a best case scenario from the perspective of system performance.

[15]Note that in the case with drop outs, miss ratio (# drop outs + # missing deadline)/total.

[16]Average tardiness for the case with drop outs cannot be computed since drop outs do not actually enter service.

[17]In our model, c-μ rule corresponds to $\delta_{ij}/(E(\textit{response-time})$ where the denominator is the expected processing time (Mendelson and Whang 1990).

Figure 4 Miss Ratio at Different Jobs Arrival Rates (λ)

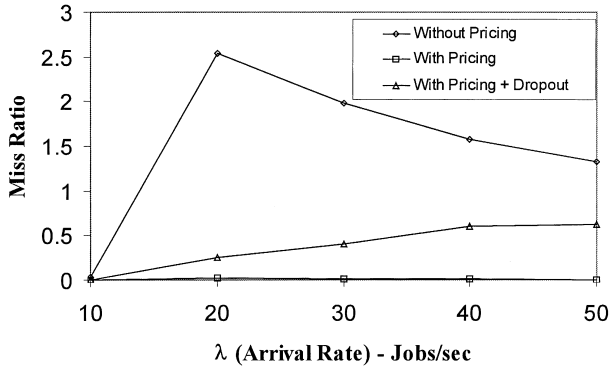


Figure 5 Average Tardiness at Different Loads

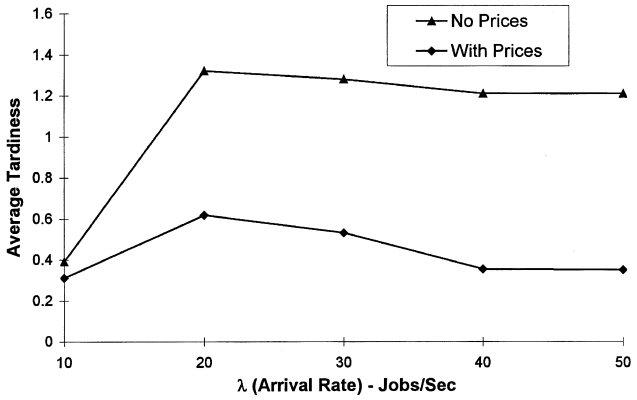


Figure 6 shows the natural logarithm of *NMR* (in %) at various multiprogramming levels for *EDF*, *LSF* and our pricing model at 20 request/second.[18] In the pricing model, multiprogramming is not required because pricing itself acts as an admission control. As expected, at low multiprogramming levels both *EDF* and *LSF* perform very well. Consistent with previous studies in RTDB, *LSF* performs better than *EDF* as the system is allowed to become overloaded (Abbott and Garcia-Molina 1992). At higher multiprogramming levels, as expected, the performance deteriorates. We now compare the economic performance of all three policies at a multiprogramming level of 60 where *NMR*s are approximately the same.[19]

[18]Natural logarithm is used to provide more resolution to the graphs.

Figure 6 **Normalized Miss Ratio at λ of 20 Jobs/Second and Different Multiprogramming Levels**

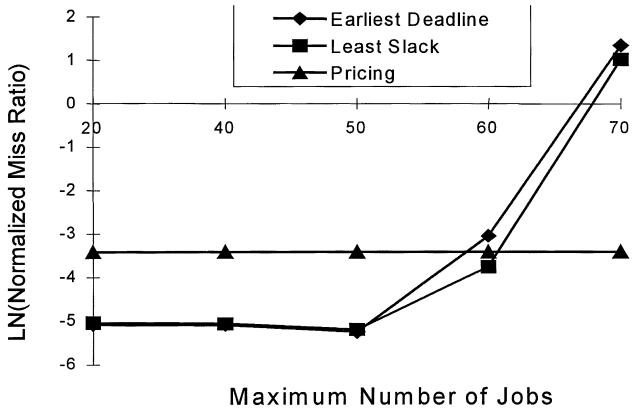


Figure 7 **Normalized Miss Ratio at λ of 50 Jobs/Second and at Different Multiprogramming Levels**

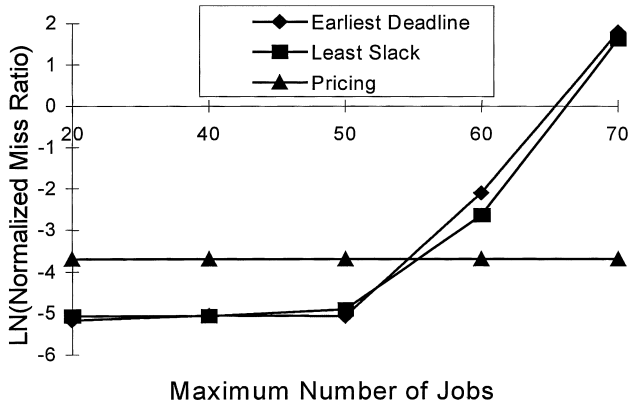


Table 4 shows the economic performance of the system at job arrival rate of 20 jobs/second. The column descriptions are self-explanatory. The last column indicates the percentage of requests that were actually admitted to the database. As the table indicates, the economic performance of pricing is far superior to *EDF* and *LSF*. Pricing performs better on two accounts: (i) the percentage of jobs submitted to the database are higher, resulting in slightly higher accumulation of instantaneous value, and (ii) the delay costs suffered are significantly smaller (approximately 12%) than that experienced with *EDF* or *LSF*. Clearly, pricing does a much better job of scheduling requests than *EDF* and *LSF*, and judiciously chooses which requests to accept for processing. Another notable statistic in the table is the consumer surplus; even though there is no charge for access with *EDF* or *LSF*, the consumers are worse off and retain only 50% of the surplus, compared to pricing.

To compare the performance of these load management approaches under heavy demand, we repeated the experiment for arrival rates of 50 jobs/sec. The logarithm of *NMR* for the three models at various multiprogramming levels is shown in Figure 7. At lower multiprogramming levels, *EDF* or *LSF* provide similar results as compared to 20 jobs/second. The reason behind this is that at lower multiprogramming levels the system is always full at both 20 and 50 jobs/second. However, at higher multiprogramming levels more requests enter the system, thus affecting the execution of all jobs. This explains the increase in *NMR* for *LSF* and *EDF*. An interesting result is that our pricing model had a lower *NMR* relative to the arrival rate of 20 jobs/second. This is consistent with our results in Experiment 1, where the miss ratio and average tardiness actually declined at higher arrival rates in our pricing scheme. The reason is that smaller requests with higher user values were accepted for processing. Requests that consumed more resources and added little to the benefits of the system were rejected. The economic performance for various priority schemes is shown in Table 5. As expected, the results indicate even higher

[19]Note that at a multiprogramming level of 60 the natural logarithm of NMR is negative, i.e., the NMR is less than 1%.

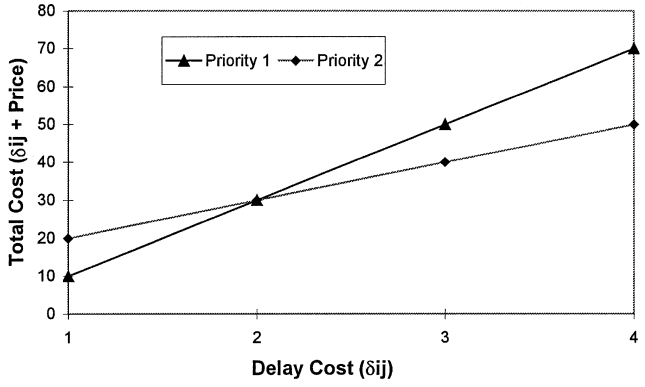
Table 4 Net Benefits and Consumer Surplus for 20 Jobs/Second

	Total Instantaneous Value per second (V)	Total Delay cost per second (D)	Total Net Benefits (W) (W = V − D)	Total Price (P)	Consumer Surplus (W − P)	Submitted (%)
EDF	316	214	102	0	102	63.3
LSF	316	213	103	0	103	63.3
Pricing	373	25	348	153	195	67.2

Table 5 Net Benefits and Consumer Surplus for 50 Jobs/Second

	Total Instantaneous Value per second (V)	Total Delay cost per second (D)	Total Net Benefits (W = V − D)	Total Price (P)	Consumer Surplus (W − P)	Submitted (%)
EDF	316	222	94	0	94	25.3
LSF	316	224	92	0	92	25.3
Pricing	563	33	530	246	283	39.5

Figure 8 Total User Cost in Different Priorities With Different δ_{ij}



benefits by using pricing as compared to that at a job arrival rate of 20 jobs/sec. The net benefits are more than five times higher and consumer surplus is almost three times higher with pricing, compared to those at *EDF* and *LSF*. The significant difference in these benefits is because with pricing, jobs with a higher ratio of value to processing requirements were admitted and, as a result, a significantly higher number of jobs were admitted (39.5% versus 25.3%).

These simulation results indicate that both economic efficiency and database performance can be improved by using a pricing mechanism for admission control and scheduling.

6. Conclusion and Future Research

In this paper, we have explored a unique approach for providing timely information services within organizations using RTDBs. We apply a priority-pricing mechanism to manage negative externalities in the operation of RTDBs as an alternative to complex congestion control and scheduling techniques suggested in the RTDB literature. Our approach maximizes organizational benefits in the presence of user delay costs and fixed database processing capacity, while improving traditional database performance metrics, such as miss ratio and average tardiness. This research has two significant components: (1) ananalytical model of the database as an economic system to generate priority-price-delay schedule that maximizes organizational benefits, and (2) adaptation of the results from the analytical model as an online admission control and

scheduling technique for real-time databases. The second part of this research is validated by simulation.

The analytical model at optimality generates the priority-price-delay schedule where each priority for a given job class is associated with an expected delay. The optimal prices are congestion tolls; that is, they are equal to the aggregate delay cost imposed on all other users of the RTDB. Therefore, prices are higher for higher priority jobs because they impose additional delay costs on lower priority jobs. This pricing mechanism controls the flow of jobs to the system to achieve a desired response time for all the jobs in the system.

The second part of this paper involves operationalization of the analytical results to manage database resources. The analytical results, based on long-run stochastic equilibrium, may not hold true for short-run increases in the arrival processes as temporary database overload may bring down the system completely. Therefore, we proposed an adaptive mechanism that incrementally adjusts the priority-price-delay schedule to reflect the true demand and the state of the system. When congestion increases, the priority-price-delay schedule is adjusted to restrict the flow of jobs. Alternatively, the schedule is adjusted to accept more jobs when the resources are idle. This has practical implications in managing a database system, as is evident from our simulation results. We compared the results from theoretically derived prices against those from traditional admission control and scheduling mechanisms suggested in the computer science literature. We compared our model against *FCFS, EDF* and *LSF* scheduling algorithms for various workload parameters. In the simulation experiments, we model the RTDB operational intricacies (such as intermediate queuing for disk and CPU accesses) that were ignored in the analytical model for tractability.

Our simulation results showed that our pricing mechanism not only maximized organizational benefits, but also improved traditional database performance measures, such as the miss ratio and the average tardiness, when compared to existing admission control and scheduling mechanisms. There were certain interesting observations in the simulation experiments. At higher arrival rates, the system admitted only those requests that had significant net benefits. On the other hand, jobs with low values that consumed significant resources were effectively blocked. Even if user jobs decided to drop out because of delays greater than expected, the net system benefits remained unchanged. The performance gain is particularly significant in overload conditions. This is not surprising, because at low arrival rates most scheduling mechanisms work equally well.

Our theoretical model is applicable only to database services within organizations where pricing is used as internal transfer pricing. To apply our model in competitive environments we need to consider several other factors. For example, pricing for commercial services must consider the issue of competition and competitive pricing strategies. In particular, competitive pricing involves niche pricing and market segmentation, not considered in this paper. There is a significant amount of research that has investigated the issue of pricing in the context of competition (Kalai et al. 1992, Li and Lee 1994, Lederer and Li 1997). Another limitation of this study is that the delay cost is assumed to increase linearly with time. Note that our results will hold qualitatively under traditional deadline oriented delay cost structures considered in the database literature, where once the deadline is reached the net benefits of the requests are zero. Furthermore, as is evident from our simulations involving drop outs, the pricing implementation seems to be quite robust with respect to variations in delay cost. Even with drop outs, the system stays at a similar benefits level.

In future studies, we will also investigate how to allocate resources in a dynamic environment for various categories of workload, such as update, triggered, and read-only transactions. We will further enhance the model to consider different sets of job values and types of delay cost structures (e.g., nonlinear delay costs). We will also investigate the issues of data replication and service location from an economic perspective. An economic transaction management is a viable option for managing resources in a database environment.

Acknowledgments. The authors are indebted to the editor, associate editor, and anonymous reviewers for their helpful comments and suggestions in improving the paper.

The work of the first author is supported by the Na-

tional Science Foundation CAREER Award under contract No. IIS-9875746.

A version of this paper appeared in the *Proceedings of the Seventeenth Annual International Conference on Information Systems (ICIS)*, December 15–18, 1996, Cleveland, Ohio. It received the Best Paper Runner-up Award at the conference. This paper is substantially revised and extended.

Appendix A: Estimation of Delay Costs

By monitoring the current state of the database (prices and waiting times), the users' delay costs can be estimated by monitoring their choices, assuming that users are rational and maximize their own utility. Consider Figure 8, where there are two priority classes, Priority 1 is associated with a price of 20 and a throughput time of 10, while Priority 2 is associated with a price of 10 and a throughput time of 20. Depending upon the δ_{ij} of users, different priorities are optimal for them. For example, in the case presented in Figure 8, users with δ_{ij} of less than or equal to one will choose Priority two and users having δ_{ij} of greater than 1 will choose Priority 1. Therefore, by monitoring the choices users make, one can create a histogram of user delay costs based on their transaction classes. These histograms can then be upgraded in a Bayesian manner by considering the historic histograms collected so far and allocating the current period distribution in accordance with the historical patterns. In the case in which there is a single priority class, similar differentiation can be made as long as the service provider has an idea of opportunity costs of the users (for example, by having price and performance information regarding a competitor). Gupta et al. (1997) provide the complete description of this approach and its effectiveness. Their results show that with the estimated average delay cost of users in each priority class, the resulting system benefits show minimal loss in efficiency.

References

Abbott, R., H. Garcia-Molina. 1988. Scheduling real-time transactions. *SIGMOD Record* **17**(1) 71–81.

——, ——. 1992. Scheduling real-time transactions: Performance evaluation. *ACM Trans. Database Systems* **17**(3) 513–560.

Agrawal, R., M. J. Carey, M. Livny. 1987. Concurrency control performance modeling: Alternatives and implications. *ACM Trans. Database Systems* **12**(4) 609–654.

Bazaraa, M. S., C. Shetty. 1993. *Nonlinear Programming: Theory and Algorithms*. Wiley, New York.

Bestavros, A. 1996. Advances in real-time database systems research. *SIGMOD Record* **25**(1) 3–7.

Branding, H., A. P. Buchmann. 1996. Unbundling RTDBMS functionality to support WWW applications. *Proc. First Internat. Workshop on Real-Time Databases,* Newport Beach, CA. 45–47.

Cocchi, R., S. Shenker, E. Ester, L. Zhang. 1993. Pricing in computer networks: Motivation, formulation, and example. *IEEE/ACM Trans. Networking* **1**(6) 614–627.

Dewan, S., H. Mendelson. 1990. User delay costs and internal pricing for a service facility. *Management Sci.* **36**(12) 1502–1517.

Giridharan, S., ——. 1994. Free access policy for internal networks. *Inform. Systems Res.* **5** 1–21.

Gupta, A., D. O. Stahl, A. B. Whinston. 1996. An economic approach to networked computing with priority classes. *Organ. Comput. Elect. Commerce* **6**(1) 71–95.

——, ——, ——. 1997a. A stochastic equilibrium model of Internet pricing. *J. Econom. Dynam. Control* **21** 697–722.

——, B. Jukic, ——, ——. 1997b. Designing an incentive compatible mechanism for Internet traffic pricing. *DIMACS Workshop on Economics, Game Theory, and the Internet.* New Brunswick, NJ. ⟨http://www.opim.uconn.edu/users/alokpapers/abda_rut.ps⟩

Hahn, F. 1982. *Stability.* K. Arrow and M. Intriligator, eds. *Handbook of Mathematical Economics, Vol. II.* North-Holland, Amsterdam.

Haritsa, J., M. Livny, M. Carey. 1991. Earliest deadline scheduling for real-time database systems. *Proc. IEEE Real-Time Systems Sympos.* San Antonio, TX. 232–242.

Huang, J., J. Stankovic, D. Towsley, K. Ramamritham. 1989. Experimental evaluation of real-time transaction processing. *Proc. IEEE Real-Time Systems Sympos.* Santa Monica, CA. 144–153.

Jhingran, A. 1996. Why commercial database systems are not real-time systems. *Proc. Workshop on Databases: Active and Real-time (Concepts Meet Practice)*, (In association with *5th Internat. Conf. Inform. Knowledge Management*. 50–53.

Kalai, E., M. Kamien, M. Rubinovitch. 1992. Optimal speeds in a competitive environment. *Management Sci.* **38** 1154–1163.

Kleinrock, L. 1967. Optimum bribing for queue position. *Oper. Res.* **15** 304–318.

Kim, W., J. Srivastava. 1991. Enhancing real time DBMS performance with multiversion data and priority based disk scheduling. *Proc. IEEE Real-Time Systems Sympos.* San Antonio, TX. 222–231.

Konana, P. 1995. A transaction model for active and real-time databases. Ph.D Thesis, The University of Arizona, Tucson, AZ.

——, A. Gupta, A. B. Whinston. 1996a. Research issues in real-time DBMS in the context of electronic commerce. *ACM Proceedings of the Workshop on Databases Active and Real-Time: Concepts Meet Practice,* Rockville, MD.

——, ——, ——. 1996b. Pricing of information services using real-time databases: A framework for integrating user preferences and real-time workload. *Proceedings of the Seventeenth International Conference on Information Systems,* Cleveland, OH.

Kurose, J. F., R. Simha. 1989. A microeconomic approach to optimal resource allocation in distributed computer systems. *IEEE Trans. Computers.* **38**(5) 705–717.

Law, A. M., W. D. Kelton. 1991. *Simulation Modeling and Analysis.* McGraw Hill, New York.

Lee, J., S. H. Son. 1993. Using dynamic adjustment of serialization order for real-time database systems. *Proc. IEEE Real-Time Systems Symposium,* Raleigh-Durham, NC. 66–75.

Lederer, P. J., L. Li. 1997. Pricing, production, scheduling, and delivery-time competition. *Oper. Res.* **45** 407–420.

Li, L., Y. S. Lee. 1994. Pricing, and delivery-time performance in a competitive environment. *Management Sci.* **40** 633–646.

MacKie-Mason, J. K., H. R. Varian. 1995. Pricing the Internet. B. Kahin, J. Keller, eds. *Public Access to the Internet.* Prentice-Hall, New York.

Mendelson, H. 1985. Pricing computer services: Queuing effects. *Comm. ACM* **28** (3) 312–321.

——, S. Whang. 1990. Priority pricing for the M/M/I queue. *Oper. Res.* **38**(5) 870–883.

Naor, P. 1969. On the regulation of queue size by levying tolls. *Econometrica* **37** 15–24.

Pang, H., M. Livny, M. J. Carey. 1992. Transaction scheduling in multi-class real-time database systems. *Proc. IEEE Real-Time Systems Sympos.* Phoenix, AZ. 23–24.

Ramamritham, K. 1993. Real-time databases. *Internat. J. Distributed and Parallel Databases* **1**(2) 199–226.

Schwetman, H. 1991. Microelectronics and Computer Technology Corporation (MCC) at Austin, TX.

Shenker, S. 1995. Service models and pricing policies for an integrated services Internet. B. Kahin, J. Keller, eds. *Public Access to the Internet.* Prentice-Hall, Englewood Cliffs, NJ.

Silberschatz, A., J. L. Peterson, P. B. Galvin. 1992. *Operating System Concepts.* Addison-Wesley Publication Company, New York.

Stankovic, J. A. 1988. Misconceptions about real-time computing. *IEEE Computer* **21**(10) 10–18.

Stahl, D., A. B. Whinston. 1994. A general equilibrium model of distributed computing. W. W. Cooper, A. B. Whinston, eds. *New Directions in Computational Economics.* Kluwer Academic Publishers, Dordrecht, The Netherlands 175–189.

Stidham, S. 1992. Pricing and capacity decisions for a service facility: Stability and multiple local optima. *Management Sci.* **38** 1121–1139.

Stonebraker, M., P. M. Aoki, A. Pfeffer, A. Sah, J. C. Staelin, A. Yu. 1996. MARIPOSA: A wide-area distributed database system. *J. VLBD* **5**(1) 64–84.

——, R. Devine, M. Kornacker, W. Litwin, A. Sah, C. Staelin. 1994. An economic paradigm for query processing and data migration in Mariposa. *Proceedings of 3rd International Conference on Parallel and Distributed Inform. Systems.* Austin, TX.

Westland, J. C. 1992. Congestion and network externalities in the short run pricing of information systems services. *Management Sci.* **38** 992–1009.

Yechiali, U. 1971. On optimal balking rules and toll charges in the GI/M/I queue process. *Oper. Res.* **19** 348–370.

——. 1972. Customers' optimal joining rules for the GI/M/s queue. *Oper. Res.* **18** 434–443.

Yu, P. S., K-L. Wu, K-J. Lin, S. H. Son. 1994. On real-time databases: Concurrency control and scheduling. *Proc. of the IEEE* **82**(1) 140–157.

Izak Benbasat, Associate Editor. This paper was received on May 27, 1997, and was with the authors 10 months for 3 revisions.