ELSEVIER

Information Sciences 111 (1998) 19–47

INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL

Analysis of concurrency control protocols for real-time database systems

Özgür Ulusoy [1]

Department of Computer Engineering and Information Sciences, Bilkent University, 06533 Bilkent, Ankara, Turkey

Received 8 May 1996; received in revised form 3 May 1997; accepted 17 August 1997

Abstract

This paper provides an approximate analytic solution method for evaluating the performance of concurrency control protocols developed for real-time database systems (RTDBSs). Transactions processed in a RTDBS are associated with timing constraints typically in the form of deadlines. The primary consideration in developing a RTDBS concurrency control protocol is the fact that satisfaction of the timing constraints of transactions is as important as maintaining the consistency of the underlying database. The proposed solution method provides the evaluation of the performance of concurrency control protocols in terms of the satisfaction rate of timing constraints. As a case study, a RTDBS concurrency control protocol, called High Priority, is analyzed using the proposed method. The accuracy of the performance results obtained is ascertained via simulation. The solution method is also used to investigate the real-time performance benefits of the High Priority over the ordinary Two-Phase Locking. © 1998 Elsevier Science Inc. All rights reserved.

Keywords: Real-time database systems; Concurrency control; Performance evaluation; Markov modeling; Analytic solution

1. Introduction

A real-time database system (RTDBS) is designed to provide timely response to the transactions of data-intensive applications. Examples of

[1] E-mail: oulusoy@bilkent.edu.tr; fax: (90)(312)266-4126.

RTDBS application areas include computer-integrated manufacturing, airline reservation systems, stock market, banking, and command and control systems. Similar to a conventional real-time system, transactions processed in a RTDBS are associated with timing constraints, typically in the form of deadlines. Access requests of transactions to data or other system resources are scheduled on the basis of the timing constraints. What makes a RTDBS different from a real-time system is the requirement of preserving the logical consistency of data in addition to considering the timing constraints of transactions. The requirement to maintain data consistency is the essential feature of a conventional database system. However, the techniques used to preserve data consistency in database systems are all based on transaction blocking and transaction restart, which makes it virtually impossible to predict computation times and hence to provide schedules that guarantee deadlines in a RTDBS. As a result, it becomes necessary to extend traditional database management techniques with time-critical scheduling methods. While the primary performance goal in conventional database systems is to minimize the average response time of transactions and to maximize throughput, the main objectives in RTDBSs is to maximize the number of transactions that satisfy their deadlines.

The scheduling problem in RTDBSs has been addressed by a number of recent studies. The general approach taken in developing new scheduling algorithms has been to use existing techniques in CPU scheduling, buffer management, IO scheduling, and concurrency control, and to apply time-critical scheduling methods to observe the timing requirements of transactions [17,30]. A considerable amount of RTDBS research has been devoted to performance evaluation of time-cognizant concurrency control protocols. However, the performance studies were either based on simulation (e.g., [1,8,10,14,15,19, 22,27,28]), or carried out on a RTDBS testbed (e.g., [13]). To the best of our knowledge, no analytic performance study has been reported so far involving the evaluation of concurrency control protocols in RTDBSs, which is the main contribution of this paper. [2]

The behavior of concurrency control protocols in traditional database systems has been investigated using both simulation (e.g., [2,9,16,18]) and analytic models (e.g., [5,6,20,21,26]). In this paper, we attempt to extend the existing analyses for concurrency control protocols to a real-time environment. [3] We analyze the performance of RTDBS concurrency control protocols via Markov modeling. As indicated in all analytic works listed above, it is practically impossible to find an exact analytic solution to the performance evaluation problem of a concurrency control protocol. To simplify the problem, we an-

[2] Recently some analytical performance studies of RTDBSs have appeared in the literature (e.g., [11,24]); however, those works do not particularly involve the performance of concurrency control protocols.

[3] Our initial work in this direction appeared in [29].

alyze an isolated individual transaction, rather than capturing the states of all concurrent transactions. It is assumed that the isolated transaction sees the average state of other transactions. This method was found to be fairly accurate in analyzing the performance of two-phase locking [5,23,25] and timestamp-ordering algorithms [20,21]. The model provided is able to reflect the impact of the presence of other transactions on the performance of the isolated transaction. However, the analysis is approximate since the average behavior of transactions is modeled rather than their dynamic behavior. The accuracy of the performance results obtained by the analysis is ascertained using a simulation program which simulates the dynamic behavior of each transaction.

The proposed solution method is used to evaluate the performance of concurrency control protocols in terms of the *satisfaction rate of timing constraints*. As a case study, we analyze a RTDBS concurrency control protocol, called *High Priority* (HP), using the solution method. Some performance experiments are conducted to evaluate the protocol under various conditions. To see how the 'real-time' aspect makes a difference, the solution method is also used to model ordinary Two-Phase Locking (2PL), and the performances of 2PL and HP are comparatively evaluated. The experiment results help us show that the proposed analytic solution is a valid and useful method to predict the performance of concurrency control protocols developed for RTDBSs.

The remainder of the paper is organized as follows. The next section provides the structure and characteristics of a RTDBS model used in the evaluation of concurrency control protocols. Section 3 describes the analytic solution method proposed to model the behavior of the protocols. In Section 4, protocol HP is described, and an analysis of the protocol based on the proposed solution method is provided. Validation results of the analytic solution and the results of some experiments are also discussed in this section. Section 5 extends the analysis by removing some of the constraints applied in developing the RTDBS model. Finally, Section 6 provides a brief discussion of our work together with the future plans.

2. RTDBS model

This section briefly describes the RTDBS model used in evaluating the performance of RTDBS concurrency control protocols. The model is based on a closed queuing model of a single site database system. It contains one CPU resource shared by transactions.

Each transaction submitted to the system is associated with a real-time constraint in the form of a deadline, and is assigned a unique real-time priority determined on the basis of its deadline. Any priority assignment policy that makes use of the deadline information can be adapted by the system. 'Earliest Deadline First' (EDF) policy is one possible candidate which states that a transaction with an earlier deadline has higher priority than a transaction with a later deadline [1]. It is assumed that if any two transactions have the same deadline, the one that has arrived at the system earlier is considered to have a higher priority. The priority of a transaction is 'static'; i.e., the priority assigned at transaction's arrival time remains the same during the lifetime of the transaction.

The 'slack time' of a transaction is defined as the distance from the current time to the deadline of the transaction. The slack time of a new transaction in our system is considered to be proportional to the estimated response time of the transaction, and the proportionality factor is determined by the parameter S. (See the derivation of slack time ST of a transaction in Section 4.1.2). While our calculations involve the estimation of transaction processing times in assigning deadlines, we assume that the system itself lacks the knowledge of processing time information. [4] It is assumed that the delay for the initialization of each transaction is distributed exponentially with mean $1/\mu_0$.

The basic unit of access (or locking) is referred to as a data item. The number of data items stored in the database is denoted by the parameter D. Concurrent data access requests of transactions are controlled by using a concurrency control protocol. Depending on its real-time priority, an access request of a transaction is either granted or results in blocking or abort of the transaction. If the access request is granted, the transaction obtains a lock on the data item and starts processing it. The processing time at the CPU is assumed to be exponentially distributed with mean $1/\mu_P$. A blocked transaction is not allowed to proceed until after the data lock it requires is released. A transaction releases all the locks it has been holding after it is either committed or aborted. A transaction can be committed after it has processed the last data item in its access list. An executing transaction is aborted if its deadline expires. Depending on the concurrency control protocol adapted, a data conflict might also lead to an abort decision.

The other primary constraints applied in developing our model are:

- Each transaction accesses the same number of data items, which is specified by the parameter d.
- Data items accessed by each transition are uniformly distributed over the database with no duplicates.
- All data accesses are exclusive (i.e., there are no shared locks).
- The shared database system is memory-resident; thus, an access to a data item does not involve any disk access.
- The transaction population in the system (the level of multiprogramming) is constant and determined by the parameter t.

[4] 'Deadline' is the only information provided by the arriving transaction to be used in scheduling decisions.

Section 5 extends the analysis of each of the following cases: variable transaction size, non-uniform data accesses, shared as well as exclusive locking modes, and a disk-resident database. A discussion on the possibility of relaxing the constant transaction population assumption is provided in the last section.

Table 1 summarizes the parameters of the RTDBS model.

3. Performance analysis of RTDBS concurrency control protocols

3.1. Steady-state distribution

All transactions processed in the system are assumed to be identical and exhibit the average steady-state behavior. The execution of an isolated individual transaction is modeled by a Markov chain with $2d+1$ states as shown in Fig. 1. State (0) of the chain represents the initialization phase of the transaction. The other $2d$ states are labeled by a tuple (i, X), where i is an integer which can take any value from the set $\{1,2,\ldots,d\}$, and denotes that the transaction is accessing its ith data item. X can take either of the two values: B (blocked) or P (processing). The access request of the transaction on a data item might result in blocking of the transaction (with probability P_b). In a RTDBS environment, the real-time priority of the transaction plays the major role in determining the blocking probability (see Section 4.1.1). State $(i$, B) represents the situation that the transaction is blocked at its attempt to access its ith data item. The blocking times of the transaction are assumed to be independent and identically distributed; the blocking delay at state $(i$, B) is assumed to be exponentially distributed with mean $1/\mu_B$, for all $i \in \{1,2,\ldots,d\}$. State $(i$, P), denotes the case that the transaction is processing its ith data item. The lock on a data item is obtained right before processing it. After processing a data item, the next data item to be accessed by the transaction is chosen from a uniform distribution among all data items that have not already been accessed by the transaction. The data conflict check for the first data access request of the transaction (which will lead to either blocking of the transaction or granting the lock on the requested data item) is performed in state (0), while that for the ith request $(2 \leqslant i \leqslant d)$ is performed before leaving the processing

Table 1
Parameters of the RTDBS model

S	Slack factor used in assigning transaction deadlines
μ_0	Mean transaction initialisation rate
μ_P	Mean CPU service rate
D	Number of data items stored in the database
d	Number of data items accessed by each transaction
t	Number of transactions processed in the system at any moment in time

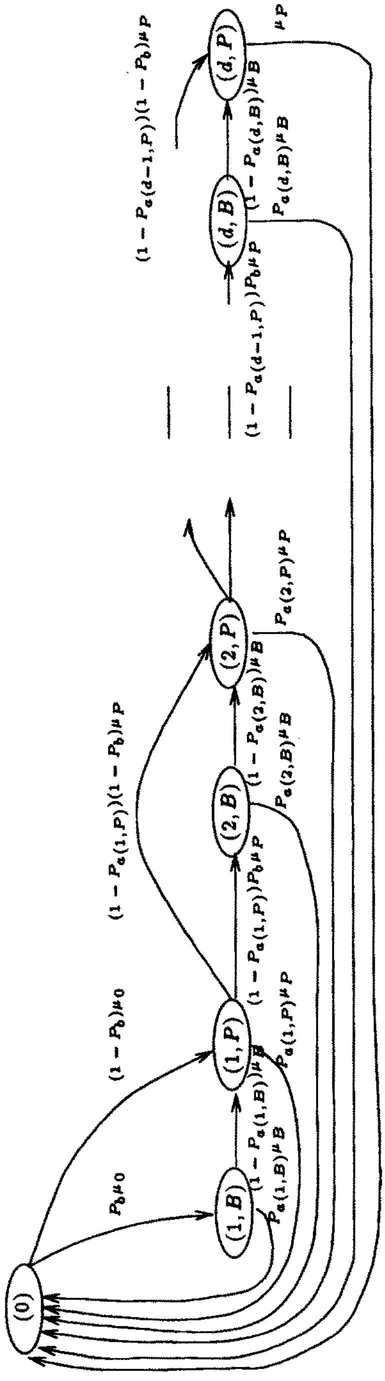

Fig. 1. State-transition diagram for a transaction.

state ($i-1$, P). At any state (i, X), it is possible that the transaction can be aborted as a result of a data conflict or due to the situation that its deadline has expired. An aborted transaction releases all the locks it has been holding. The aborting probabilities in states (i, B) and (i, P) are denoted by $P_{\mathrm{a}(i,B)}$ and $P_{\mathrm{a}(i,P)}$, respectively. It is assumed that aborting a transaction at any state does not take effect until the transaction leaves that state. An aborted transaction goes to state (0) to be reinitialized and it returns to the system as a new transaction. As discussed before, the number of transactions executing in the system at any moment in time is kept constant.

When the transaction completes processing d data items, it is said to be committed and it goes to state (0) to be initialized as a new transaction. A transaction cannot be aborted after processing its last data item; i.e., $P_{\mathrm{a}(d,\mathrm{P})} = 0$.

Let $\{\mathbf{P}(0), \mathbf{P}(1,\mathrm{B}), \mathbf{P}(1,\mathrm{P}), \mathbf{P}(2,\mathrm{B})\mathbf{P}(2,\mathrm{P}), \ldots, \mathbf{P}(d,\mathrm{B}), \mathbf{P}(d,\mathrm{P})\}$ be the steady-state distribution of the Markov chain (each element in this set denotes the probability of being at a particular state). The following system of linear equations can be given for this distribution:

$$\begin{aligned}
\mathbf{P}(1,\mathrm{B}) &= \frac{\mu_0}{\mu_\mathrm{B}} P_\mathrm{b}\mathbf{P}(0),\\
\mathbf{P}(1,\mathrm{P}) &= \frac{\mu_0}{\mu_\mathrm{P}}(1 - P_\mathrm{b}P_{\mathrm{a}(1,\mathrm{B})})\mathbf{P}(0),\\
\mathbf{P}(2,\mathrm{B}) &= \frac{\mu_0}{\mu_\mathrm{B}} P_\mathrm{b}(1 - P_{\mathrm{a}(1,\mathrm{P})})(1 - P_\mathrm{b}P_{\mathrm{a}(1,\mathrm{B})})\mathbf{P}(0),\\
\mathbf{P}(2,\mathrm{P}) &= \frac{\mu_0}{\mu_\mathrm{P}}(1 - P_\mathrm{b}P_{\mathrm{a}(1,\mathrm{B})})(1 - P_{\mathrm{a}(1,\mathrm{P})})(1 - P_\mathrm{b}P_{\mathrm{a}(2,\mathrm{B})})\mathbf{P}(0),\\
\mathbf{P}(3,\mathrm{B}) &= \frac{\mu_0}{\mu_\mathrm{B}} P_\mathrm{b}(1 - P_{\mathrm{a}(1,\mathrm{P})})(1 - P_\mathrm{b}P_{\mathrm{a}(1,\mathrm{B})})(1 - P_{\mathrm{a}(2,\mathrm{P})})\\
&\quad \times (1 - P_{\mathrm{b}P_\mathrm{a}(2,\mathrm{B})})\mathbf{P}(0),\\
\mathbf{P}(3,\mathrm{P}) &= \frac{\mu_0}{\mu_\mathrm{P}}(1 - P_\mathrm{b}P_{\mathrm{a}(1,\mathrm{B})})(1 - P_{\mathrm{a}(1,\mathrm{P})})(1 - P_\mathrm{b}P_{\mathrm{a}(2,\mathrm{B})})\\
&\quad \times (1 - P_{\mathrm{a}(2,\mathrm{P})})(1 - P_\mathrm{b}P_{\mathrm{a}(3,\mathrm{B})})\mathbf{P}(0),\\
&\ \ \vdots
\end{aligned}$$

$$\mathbf{P}(i,\mathrm{B}) = \frac{\mu_0}{\mu_\mathrm{B}} P_\mathrm{b}\prod_{k=1}^{i-1}[(1 - P_{\mathrm{a}(k,\mathrm{P})})(1 - P_\mathrm{b}P_{\mathrm{a}(k,\mathrm{B})})]\mathbf{P}(0), \quad i \in \{1,2,\ldots,d\}, \tag{1}$$

$$\mathbf{P}(i,\mathrm{P}) = \frac{\mu_0}{\mu_\mathrm{P}}(1 - P_\mathrm{b}P_{\mathrm{a}(1,\mathrm{B})})\prod_{k=1}^{i-1}[(1 - P_{\mathrm{a}(k,\mathrm{P})})(1 - P_\mathrm{b}P_{\mathrm{a}(k+1,\mathrm{B})})]\mathbf{P}(0), \quad i \in \{1,2,\ldots,d\}, \tag{2}$$

$$\mathbf{P}(0) + \sum_{i=1}^{d}(\mathbf{P}(i,\mathrm{B}) + \mathbf{P}(i,\mathrm{P})) = 1. \tag{3}$$

Table 2
Probability ($P_{\mathrm{COMMIT}|(i,X)}$) of committing, given that the current state is (i,X)

X	$P_{\mathrm{COMMIT}(i,X)}$
B	$(1-P_{\mathrm{a}(i,\mathrm{B})})\prod_{k=i}^{d-1}[(1-P_{\mathrm{a}(k,\mathrm{P})})(1-P_{\mathrm{b}}P_{\mathrm{a}_{(k+1,\mathrm{B})}})]$
P	$(1-P_{\mathrm{a}(i,\mathrm{P})})\prod_{k=i+1}^{d}[(1-P_{\mathrm{b}}P_{\mathrm{a}_{(k,\mathrm{B})}})(1-P_{\mathrm{a}(k,\mathrm{P})})]$

The system can be solved by first determining $\mathbf{P}(0)$ by substituting Eqs. (1) and (2) in Eq. (3), and then computing the other steady-state probabilities $\mathbf{P}(i,\mathrm{B}),\mathbf{P}(i,\mathrm{P})(1\leqslant i\leqslant d)$ from Eqs. (1) and (2). [5] However, the solution to each of these probabilities is provided in terms of $\mu_{\mathrm{B}},P_{\mathrm{b}},P_{\mathrm{b}(i,\mathrm{B})}$, and $P_{\mathrm{a}(i,\mathrm{P})}$. The blocking and abort probabilities can be determined on the basis of the concurrency control protocol employed. Computation of variables $P_{\mathrm{b}},P_{a(i,\mathrm{B})}$, and $P_{\mathrm{a}(i,\mathrm{P})}$ is provided for the High Priority protocol in Sections 4.1.1 and 4.1.2, and for the Two-Phase Locking protocol in Section 4.2.2. The average blocking time of a transaction is formulated in the next section.

3.2. Computation of average blocking delay

When a transaction T is blocked by another transaction T' on a data item, T is not reactivated until after T' releases the lock on that item (i.e., until T' is committed or aborted). The time period transaction T remains blocked is determined by the remaining lifetime of blocking transaction T' and is independent of the current state of T. [6] In estimating the average remaining lifetime of the blocking transaction, we use the same steady-state distribution and other probabilities as the isolated transaction, because all transactions in the system are assumed to be identical and exhibit the average steady-state behavior.

Given that the current state of a transaction is (i,X), the average remaining time $\mathrm{RT}_{(iX)}$ of the transaction can be determined by the following formula:

$$\mathrm{RT}_{(iX)} = P_{\mathrm{COMMIT}|(i,X)}D_{(i,X);\mathrm{COMMIT}} + \sum_{(j,Y)=(i,X)}^{d,\mathrm{P}} \left(P_{\mathrm{a}(j,Y)|(i,X)}D_{(i,X);(j,Y)}\right), \qquad (4)$$

where $P_{\mathrm{COMMIT}|(i,X)}$ is the probability that the transaction will commit given that its current state is (i,X) (See Table 2), the implicit assumption in the formulas presented is $\prod_{i=a}^{b}f(i)=1$, if $a>b$; $D_{(i,X);\mathrm{COMMIT}}$ is the average time distance between state (i,X) and the commit time (see Table 3); $P_{\mathrm{a}(j,Y)|(i,X)}$ is the probability that the transaction will be aborted in state (j,Y) given that its current state is (i,X) (see Table 4); and $D_{(i,x);(j,Y)}$ is the average time distance from state

[5] Note that, the equations assume $\prod_{i=1}^{b}f(i)=1$, if $a>b$.
[6] Assuming that the number of locks held by the transaction $\ll D$ (database size).

Table 3
Average distance ($D_{(i,X);\text{COMMIT}}$) from state (i,X) to commit

X	$D_{(i,X);\text{COMMIT}}$
B	$\frac{1}{\mu_P} + (d-i)(P_b\frac{1}{\mu_B} + \frac{1}{\mu_P})$
P	$(d-i)(P_b\frac{1}{\mu_B} + \frac{1}{\mu_P})$

Table 4
Probability ($P_{(j,Y)\|(i,X)}$) of aborting in state (j,Y), given that the current state is (i,X)

X	Y	$P_{a(j,Y)\|(i,X)}$	
B	B	$1 - P_{a(i,B)})(1 - P_{a(i,P)})\prod_{k=i+1}^{j-1}[(1 - P_b P_{a(k,B)})(1 - P_{a(k,P)})]P_b P_{a(j,B)}$	if $j > i$
		$P_{a(j,B)}$	otherwise $(j = i)$
B	P	$(1 - P_{a(i,B)})\prod_{k=i}^{j-1}[(1 - P_{a(k,P)})(1 - P_b P_{a_{(k+1,B)}})]P_{a(j,P)}$	
P	B	$1 - P_{a(i,P)}\prod_{k=i+1}^{j-1}[(1 - P_b P_{a(k,B)})(1 - P_{a(k,P)})]P_b P_{a(j,B)}$	if $j > i$
		Undefined	otherwise
P	P	$\prod_{k=i}^{j-1}[(1 - P_{a(k,P)})(1 - P_b P_{a(k+1,B)})]P_{a(j,P)}$	

(i,X) to state (j,Y) (see Table 5). Remember that abort of a transaction in a state takes place once the transaction leaves that state. As discussed in the preceding section, it is assumed that a transaction that has just completed processing its last data item cannot be aborted (i.e., $P_{a(d,P)} = 0$).

Using the average remaining lifetime of the blocking transaction, the average time in a blocked state is estimated as

$$\frac{1}{\mu_B} = \mathbf{P}(1,\mathrm{P})\mathrm{RT}_{(1,P)} + \mathbf{P}(2,\mathrm{B})\mathrm{RT}_{(2,B)} + \mathbf{P}(2,\mathrm{P})\mathrm{RT}_{(2,P)} + \cdots + \mathbf{P}(d,\mathrm{B})\mathrm{RT}_{(d,B)} + \mathbf{P}(d,\mathrm{P})\mathrm{RT}_{(d,P)}.$$

The set of states the blocking transaction can be in excludes state $(1,\mathrm{B})$, since a blocking transaction must be holding at least one lock. The average blocking time formula can be rewritten as

Table 5
Average distance ($D_{(i,X);(j,Y)}$) from state (i,X) to state (j,Y)

X	Y	$(D_{(i,X);(j,Y)})$	
B	B	$(j-i)(\frac{1}{\mu_P} + P_b\frac{1}{\mu_B})$	
B	P	$\frac{1}{\mu_P} + (j-i)(P_b\frac{1}{\mu_B} + \frac{1}{\mu_P})$	
P	B	$(P_b\frac{1}{\mu_B} + (j-i-1)(\frac{1}{\mu_P} + P_b\frac{1}{\mu_B})$	if $j > i$
		Undefined	otherwise $(j = i)$
P	P	$(j-i)(P_b\frac{1}{\mu_B} + \frac{1}{\mu_P})$	

$$\frac{1}{\mu_B} = \sum_{(i,X)=(1,P)}^{(d,P)} (\mathbf{P}(i,X)\mathrm{RT}_{(i,X)}). \tag{5}$$

The effects of chained blockings is reflected in this formula, since the calculation of the remaining time (which determines the length of blocking delay) takes the delay of blockings into account. The computation of μ_B requires numerical iteration as will be detailed in Section 4.1.3.

3.3. Performance metric

We are primarily interested in the rate a transaction satisfies its deadline. The transaction completion rate will be a good performance measure because a transaction makes its deadline if and only if it completes processing all data items in its access list (late transactions are aborted). The completion (commit) rate γ of a transaction can be computed from the steady-state distribution of the system

$$\gamma = \mathbf{P}(d, P)\mu_P.$$

4. Case study: Modeling and an evaluation of the high priority protocol

'High Priority' (HP), described in [1], is one of the most popular RTDBS concurrency protocols proposed so far. Protocol HP is based on the two-phase locking scheme, and it aborts a low priority transaction when one of its locks is requested by a higher priority transaction. HP is characterized by its simplicity and low implementation overhead compared to the other RTDBS concurrency control protocols appeared in the literature. Also, in an earlier simulation work, we found that it can outperform other protocols under a variety of execution environments [28]. Although in this paper we concentrate on the evaluation of protocol HP, the ideas presented are also applicable to the analysis of other concurrency control protocols developed for RTDBSs.

In protocol HP, the winner in the case of a lock conflict between two transactions is always the higher priority transaction. In resolving a conflict, if the transaction requesting the lock has higher priority than the transaction that holds the lock, the latter transaction is aborted and the lock is granted to the former one. Otherwise, the lock-requesting transaction is blocked by the higher priority lock-holding transaction.

A high priority transaction never waits for a lower priority transaction. This condition prevents deadlocks if it is assumed that the real-time priority of a transaction does not change during its lifetime and that no two transactions have the same priority.

4.1. Modeling protocol HP

4.1.1. Computation of blocking probability

P_b is the probability of blocking the transaction at its data access attempt at any point of its execution. We assume that this probability is independent of the current state and the past history of the transaction (i.e., the number of data locks held by the transaction). This assumption is reasonable as long as $D \gg d$. P_b is estimated by using the following formula:

$$P_b = \frac{\text{Locks\_hp}}{D}.$$

Locks_hp stands for the average number of locks held by transactions with *higher priority*. The number of transactions that have higher priorities than the priority of the isolated transaction can be $0, 1, 2, \ldots, (t-1)$ with equal probability. That is, the average number of transactions with higher priorities will be $(0 + 1 + 2 + \cdots + (t-1))/t = (t-1)/2$. Let L denote the average number of data items locked by a transaction. L can be formulated as a function of the steady-state distribution.

$$L = \sum_{i=1}^{d} [(i-1)\mathbf{P}(i, \mathrm{B}) + i\mathbf{P}(i, \mathrm{P})]. \tag{6}$$

Note that, the number of locks held by the transaction in state (i, B) is $i - 1$, while that number is i in state (i, P). Based on these observations, we may write

$$\text{Locks\_hp} = \frac{(t-1)L}{2}. \tag{7}$$

P_b can then be expressed as

$$P_b = \frac{(t-1)L}{2D}. \tag{8}$$

4.1.2. Computation of abort probabilities

The transaction can be aborted at any state (i, X) (where $i \in \{1, 2, \ldots, d\}$, and $X \in \{\mathrm{B}, \mathrm{P}\}$ due to any of the following two facts:

- a data conflict occurs (i.e., one of its locks is requested by a higher priority transaction),
- deadline of the transaction expires.

Thus, two separate components, $P_{a(i,X)}(1)$ and $P_{a(i,X)}(2)$, are involved in the evaluation of the abort probability at any state.

$$P_{a(i,\mathrm{B})} = P_{a(i,\mathrm{B})}(1) + P_{a(i,\mathrm{B})}(2) - P_{a(i,\mathrm{B})}(1) * P_{a(i,\mathrm{B})}(2), \quad i \in \{1, 2, \ldots, d\}, \tag{9}$$

$$\begin{aligned} P_{a(i,\mathrm{P})} &= P_{a(i,\mathrm{P})}(1) + P_{a(i,\mathrm{P})}(2) \\ &\quad - P_{a(i,\mathrm{P})}(1) * P_{a(i,\mathrm{P})}(2), \quad i \in \{1, 2, \ldots, d-1\}, \end{aligned} \tag{10}$$

where $P_{a(i,B)}(1)$: The probability that the transaction will abort at blocking state (i, B) due to a data conflict. $P_{a(i,B)}(2)$: The probability that the transaction will abort at blocking state (i, B) due to expiration of its deadline. $P_{a(i,P)}(1)$: The probability that the transaction will abort at processing state (i, P) due to a data conflict. $P_{a(i,P)}(2)$: The probability that the transaction will abort at processing state (i, P) due to expiration of its deadline. As stated before, a transaction cannot be aborted after processing its last data item; i.e., $P_{a(d,P)} = 0$.

The average data access rate of a transaction is $1/(P_b(1/\mu_B) + 1/\mu_P)$ (data items per unit time). The average data access rate of all the transactions that have higher priority than that of the isolated transaction is $(t-1)/2(P_b(1/\mu_B) + 1/\mu_P)$. Therefore, the average number of data items that are accessed by all higher priority transactions during the blocking delay $1/\mu_B$ of the transaction is $(t-1)/2\mu_B(P_b(1/\mu_B) + 1/\mu_P)$. Since the transaction in state (i, B) holds $i-1$ data locks, we can specify the probability that one of the locks held by the transaction is requested by a higher priority transaction as

$$\begin{aligned} P_{a(i,B)}(1) &= \frac{(i-1)}{D}\frac{(t-1)}{2\mu_B(P_b(1/\mu_B) + 1/\mu_P)} \\ &= \frac{\mu_P}{P_b\mu_P + \mu_B}\frac{(i-1)(t-1)}{2D}. \end{aligned} \tag{11}$$

The same probability at a processing state can be specified in a similar way; however, in this case, the number of locks held by the transaction in state (i, P) is i.

$$P_{a(i,P)}(1) = \frac{\mu_B}{P_b\mu_P + \mu_B}\frac{i(t-1)}{2D}. \tag{12}$$

It is assumed that D is assigned a value large enough to produce a sensible value for the probabilities (i.e., a value within the range [0,1]).

In calculating the probability of transaction abort due to deadline expiration we employ the following approach. First, it is assumed that each transaction is assigned a deadline proportional to its size (i.e., the number of data accesses required by the transaction). The slack time ST of a new transaction (i.e., the time distance to its deadline) in our model is estimated as

$$\mathrm{ST} = S * \mathrm{RES} = S\left(\frac{1}{\mu_0} + d\left(P_b\frac{1}{\mu_B} + \frac{1}{\mu_P}\right)\right),$$

where S is the slack factor and RES is the average transaction response time. Then, denoting the average age of a transaction in state (i,X) by $\mathrm{AGE}_{(i,X)}$,

$$P_{a(i,B)}(2) = \frac{\mathrm{AGE}_{(i,B)}}{\mathrm{ST}},$$

$$P_{a(i,P)}(2) = \frac{\mathrm{AGE}_{(i,P)}}{\mathrm{ST}},$$

where

$$\mathrm{AGE}_{(i,\mathrm{B})} = \frac{1}{\mu_0} + (i-1)\left(P_\mathrm{b}\frac{1}{\mu_\mathrm{B}} + \frac{1}{\mu_\mathrm{P}}\right) + \frac{1}{\mu_\mathrm{B}},$$

$$\mathrm{AGE}_{(i,\mathrm{P})} = \frac{1}{\mu_0} + i\left(P_\mathrm{b}\frac{1}{\mu_\mathrm{B}} + \frac{1}{\mu_\mathrm{P}}\right).$$

Substitution of the average age and slack time parameters yields

$$P_{\mathrm{a}(i,\mathrm{B})}(2) = \frac{1/\mu_0 + (i-1)(P_\mathrm{b}/\mu_\mathrm{B} + 1/\mu_\mathrm{P}) + 1/\mu_\mathrm{B}}{S(1/\mu_0 + d(P_\mathrm{b}/\mu_\mathrm{B} + 1/\mu_\mathrm{P}))}, \tag{13}$$

$$P_{\mathrm{a}(i,\mathrm{P})}(2) = \frac{1/\mu_0 + i(P_\mathrm{b}/\mu_\mathrm{B} + 1/\mu_\mathrm{P})}{S(1/\mu_0 + d(P_\mathrm{b}/\mu_\mathrm{B} + 1/\mu_\mathrm{P}))}. \tag{14}$$

Abort probabilities $P_{\mathrm{a}(i,\mathrm{B})}$ and $P_{\mathrm{a}(i,\mathrm{P})}$ can be expressed in their final forms by substituting Eqs. (11)–(14) in Eqs. (9) and (10).

4.1.3. Numerical solution

Fig. 2 presents the procedure employed in solving the linear system of equations for the steady-state distribution (i.e., Eqs. (1)–(3)), average blocking delay (i.e., Eq. (5), blocking probability (i.e., Eq. (8), and aborting probabilities (i.e., Eqs. (9) and (10)). As mentioned before, a numerical iteration is needed in computing the value of the average block delay $(1/\mu_\mathrm{B})$ because a choice for μ_B determines the steady-state probabilities which when substituted in Eq. (5) generates a new computed value for μ_B.

It was observed in our initial experiments that under any set of reasonable parameter values, when the parameter ϵ of iteration is set to 0.001, the number of iterations to reach convergence never exceeds 4 with different initial values of μ_B and P_b. In the computations of the following experiments, we used an initial average blocking delay $(1/\mu_\mathrm{B})$ value of $d/2\mu_\mathrm{P}$, which corresponds to the average remaining lifetime of a transaction in a system with no contention. The blocking probability P_b was initially assumed to be $(t-1)d/4D$ by setting L (average number of locks held by a transaction) to $d/2$ in Eq. (8).

```
solution_procedure {
  $\mu'_B = 0$
  initialize $P_b$, $\mu_B$
  while $(\frac{|\mu_B - \mu'_B|}{\mu_B} > \epsilon)$ {
        Compute $P_{a(i,X)} = P_{a(i,X)}(P_b, \mu_B)$, $i \in \{1,2,...,d\}; X \in \{B,P\}$
        Compute $\mathbf{P}(0) = \mathbf{P}(0)(P_b, \mu_B, P_{a(j,Y)})$,
                $\mathbf{P}(i,X) = \mathbf{P}(i,X)(P_b, \mu_B, P_{a(j,Y)})$
                $i,j \in \{1,2,...,d\}; X,Y \in \{B,P\}$
        $P_b = P_b(\mathbf{P}(0), \mathbf{P}(i,X))$, $i \in \{1,2,...,d\}; X \in \{B,P\}$
        $\mu'_B = \mu_B$
        $\mu_B = \mu_B(\mu'_B, \mathbf{P}(0), \mathbf{P}(i,X), P_b, P_{a(j,Y)})$, $i,j \in \{1,2,...,d\}; X,Y \in \{B,P\}$
  }
}
```

Fig. 2. System solution by numerical iteration.

4.1.4. Validation results

In this section, we use the results of a simulation program to ascertain the accuracy of the analytic solution. Primary restrictions of the analytic model were removed in the simulation program. The analytic model assumed that all transactions processed in the system are identical and exhibit the average steady-state behavior. In simulation, each transaction in the system was simulated individually. The dynamic behavior of each transaction was simulated as compared to the average behavior captured by the analytic model. Due to the space limitation, the readers are referred to [28] for the details of the simulator.

Each simulation run was continued until 5000 transactions were successfully committed. The 'independent replication' method was used to validate the results by running each configuration 25 times with different random number seeds and using the averages of the replica means as final estimates. 90% confidence intervals were obtained for the simulation results. The formula involved in calculating the performance metric of interest is

$$\gamma = \frac{\text{number of committed transactions}}{\text{simulated time}}.$$

Remember that the transaction completion (commit) rate γ also specifies the rate a transaction satisfies its deadline (Section 3.3). The results of the simulation and analytic solution were compared for various sets of parameter values. In this paper, only a sample of validation results is presented for conciseness. [7] The slack factor value chosen for the estimations is $S = 5$.

Table 6 presents the performance results of the concurrency control protocol in terms of γ for both the analytic solution and simulation. The results are presented for three different values of database size (D). The level of multiprogramming (t) was set to 5,15 and 25 for each D value explored. The number of data items accessed by each transaction (d) was varied from 5 to 15 for each setting of D and t. These ranges of parameter values enabled us to observe how well the results from the analytic solution and simulation agree under both low and high levels of data contention. Each simulation result provided is the midpoint of a 4% confidence interval. The estimates for the analytic solution and simulation are quite close. The difference between the performance results obtained with the analytic solution and simulation does not exceed 10% for each setting of the parameters. The accuracy of the analytic solution is a litter better under low data conflict conditions (i.e., for low values of parameters t and d, and high values of D) compared to the accuracy obtained with high levels of data conflict. This can be attributed to the fact that the behavior of transactions is more predictable when there is not many conflicts between them.

[7] The validation results obtained for other settings of parameters are qualitatively in agreement with those presented here.

Table 6
The deadline satisfaction rate γ (transaction/second) obtained with the analytic solution and simulation with various values of D, t, and d

t	d	$D = 1000$		$D = 5000$		$D = 1000$	
		Analytic	Simulation	Analytic	Simulation	Analytic	Simulation
5	5	11.62	12.25	11.83	12.45	11.85	12.40
	7	7.26	7.66	7.53	7.90	7.57	7.88
	9	4.78	5.12	5.09	5.35	5.13	5.30
	11	3.24	3.45	3.57	3.76	3.62	3.79
	13	2.24	2.39	2.57	2.75	2.61	2.79
	15	1.56	1.71	1.87	2.03	1.92	2.10
15	15	11.02	11.49	11.70	12.17	11.79	12.08
	7	6.48	6.79	7.36	7.66	7.48	7.80
	9	3.94	4.22	4.90	5.21	5.03	5.28
	11	2.41	2.63	3.36	3.63	3.51	3.61
	13	1.48	1.62	2.35	2.52	2.50	2.66
	15	0.89	0.96	1.67	1.82	1.81	1.92
25	5	10.45	10.93	11.58	12.21	11.73	12.33
	7	5.79	6.16	7.19	7.48	7.40	7.72
	9	3.25	3.52	4.71	4.94	4.94	5.20
	11	1.81	1.96	3.17	3.41	3.40	3.66
	13	0.98	1.05	2.16	2.32	2.40	2.53
	15	0.52	0.56	1.49	1.63	1.71	1.83

4.2. An evaluation of protocol HP

4.2.1. Sample performance results

In this section, we present the results of some experiments that evaluate the performance of the High Priority protocol in terms of transaction completion rate γ (the rate a transaction satisfies its deadline) using the proposed analytic solution method. We do not aim to provide a complete set of experiments or a detailed performance study; instead, our intention is to present some examples of employing our analytic method in the evaluation of the protocol and to show that the method is capable of producing reasonable results. The average service time for processing a data item (i.e., $1/\mu_P$) and the average delay for transaction initialization (i.e., $1/\mu_0$) were both set to 10 m. The size of the database chosen for the first two experiments was $D = 1000$ data items. With the small database size value it was aimed to evaluate the protocol under high levels of data conflicts among transactions. This small database can be considered as the most frequently accessed fraction of a larger database. Calculations in all experiments were performed under three different multiprogramming levels; i.e., $t = 5, 15$, and 25 transactions.

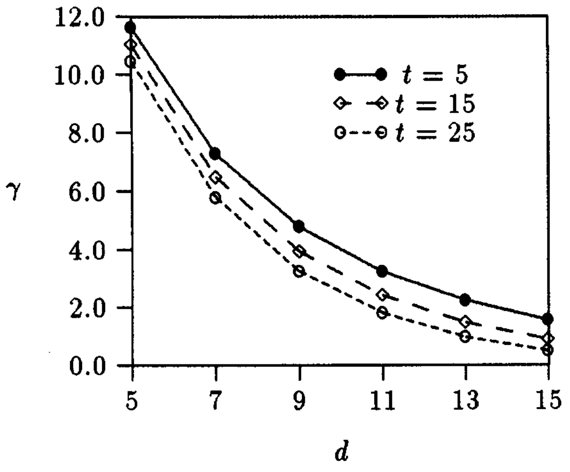

Fig. 3. Deadline satisfaction rate as a function of the transaction size.

The first experiment investigated the impact of varying average transaction size on the performance of the High Priority protocol. The parameter d was varied from 5 to 15 in steps of 2. The slack factor value used for this analysis was $S = 5$. Increasing the size of transactions corresponds to increasing number of conflicts among the concurrent transactions. As displayed in Fig. 3, the transaction completion rate (or equivalently, the deadline satisfaction rate) decreases drastically as the number of data items accessed by each transaction increases. The solution method yields sensible results because increasing number of data conflicts leads to an increase in both blocking delays and the number of conflict aborts; thus, it is likely that more transactions will miss their deadlines.

In the second experiment the value of parameter d was fixed at 10, and the effects of deadline distribution on the performance of the protocol was evaluated. The value of the slack factor parameter S was varied from 2 to 10. A small value of S corresponds to a tight deadline. Not surprisingly, the performance of the protocol becomes better as the assigned deadlines get looser. Also, the differences between the performances obtained with different multiprogramming levels increase in favor of low multiprogramming levels as the deadlines becomes larger. The results of this experiment are presented in Fig. 4.

In the last experiment, the database size D was varied from 500 to 3000 data items. The results are displayed in Fig. 5. The number of data items accessed by each transaction was $d = 10$ in this analysis. As the database size gets larger, the performance in terms of the deadline satisfaction rate becomes better for each setting of multiprogramming level t. Since the data accesses for each transaction are uniformly distributed over the database, the access sets of concurrent transactions do not have many common items when a large number of data items is stored in the underlying database; i.e., the data contention level

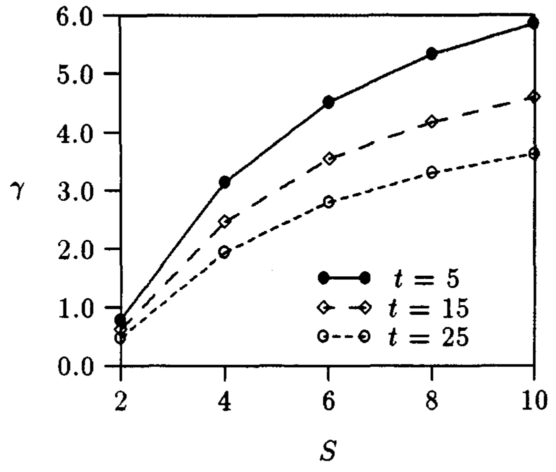

Fig. 4. Deadline satisfaction rate as a function of the slack factor that is used in assigning deadline to a new transaction.

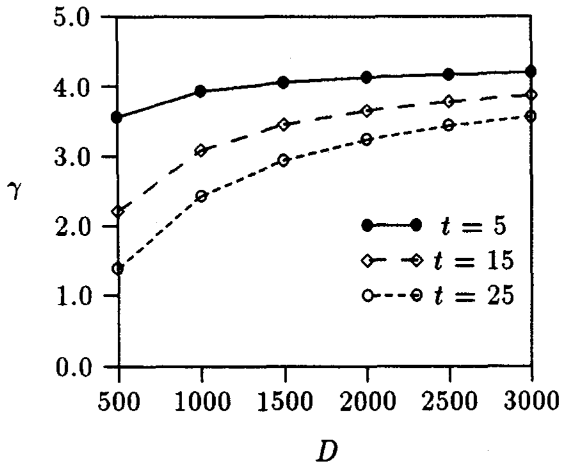

Fig. 5. Deadline satisfaction rate as a function of the database size.

in the system is low. The worse performance for small values of database size is an expected result of more data contention due to data access conflicts. Under high multiprogramming levels, the database size becomes more effective in determining the real-time performance, as can be seen from the figure.

4.2.2. Evaluating the performance improvement over the two-phase locking protocol

As described at the beginning of this section, the High Priority protocol HP extends the basic Two-Phase Locking (2PL) protocol by involving real-time priorities of transactions in scheduling decisions. To see how the 'real-time' aspect makes a difference, it would be interesting to have comparative performance results of HP and 2PL. In this section, our analytical solution method is used to model 2PL and to obtain a new set of results to be compared against the results of HP.

In 2PL protocol, a transaction is blocked on its lock request on a data item if the data item has already been locked. The transaction remains blocked until the transaction holding that lock is either committed or aborted. There is no priority aborts in this case. Deadlock is a possibility in 2PL, and whenever a deadlock occurs, one of the transaction in the deadlock cycle is aborted to resolve the deadlock. The blocking and abort probabilities of a transaction need to be recalculated for 2PL.

Computation of blocking probability for 2PL:

$$P_b = \frac{\text{Locks}}{D}.$$

Locks denotes the average number of locks held by all the transactions in the system except the isolated transaction. Remember that, in calculating the blocking probability for protocol HP, we only considered the locks held by higher priority transactions since a transaction can be blocked only if the lock requested by the transaction is currently being held by a higher priority transaction. In 2PL, no priority information is involved in scheduling, and a transaction is blocked if the requested lock is being held by any other transaction in the system.

P_b can then be expressed in terms of the average number of data items locked by a transaction (i.e., L; see Eq. (6)).

$$P_b = \frac{(t-1)L}{D}.$$

Computation of abort probabilities for 2PL:

We do not have data conflict aborts (i.e., priority aborts) in 2PL; on the other hand, a transaction can be aborted due to either a blocking deadlock or expiration of its deadline.

Deadlock checks are performed at the end of the processing states. Following any processing state (i, P) with $i = 1, 2, \ldots, d-1$, if the next data access request leads to blocking, [8] then deadlock detection has to be performed. In the case of a deadlock, the transaction is aborted to resolve the deadlock and a transition to state (0) occurs.

Abort at a blocking state (i, B), with $i = 1, 2, \ldots, d$, is possible only due to the fact that deadline of the transaction expires. An expression for the prob-

[8] At the first data access attempt there is no possibility of deadlock (even if the transaction is blocked) since no data locks is currently being held by the transaction.

ability that the transaction will abort at a blocking state due to expiration of its deadline is provided in Eq. (13) of Section 4.1.2.

The abort probability at a processing state (i, P) will be expressed in terms of two separate components $P_{\mathrm{a}(i,\mathrm{P})}(1)$ and $P_{\mathrm{a}(i,\mathrm{P})}(2)$:

$$P_{\mathrm{a}(i,\mathrm{P})} = P_{\mathrm{a}(i,\mathrm{P})}(1) + P_{\mathrm{a}(i,\mathrm{P})}(2) - P_{\mathrm{a}(i,\mathrm{P})}(1) * P_{\mathrm{a}(i,\mathrm{P})}(2), \quad i \in \{1, 2, \ldots, d-1\},$$

where $P_{\mathrm{a}(i,\mathrm{P})}(1)$ is the probability that the transaction will abort following processing state (i, P) due to a blocking deadlock, and $P_{\mathrm{a}(i,\mathrm{P})}(2)$ is the probability that the transaction will abort at the end of processing state (i, P) due to expiration of its deadline. $P_{\mathrm{a}(i,\mathrm{P})}(2)$ is specified in terms of the system parameters in Eq. (14) of Section 4.1.2. $P_{\mathrm{a}(i,\mathrm{P})}(1)$ may be expressed as

$$P_{\mathrm{a}(i,\mathrm{P})}(1) = P_{\mathrm{b}} * P_{\mathrm{dl}(i)},$$

where $P_{\mathrm{dl}(i)}$ is the probability of deadlock, given that the isolated transaction (say T) has been blocked at its attempt to obtain the $i+1st$ lock. This probability is equal to the probability that the transaction holding the requested lock (say T') is in a blocked state and it has been blocked on a data item locked by transaction T:

$$P_{\mathrm{dl}(i)} = \left(\sum_{j=2}^{d} \mathrm{P}(j, \mathrm{B}) \right) \frac{i}{(t-1)L},$$

i is the number of locks currently held by transaction T and $(t-1)L$ is the average number of locks held by all the transactions in the system except T'. In the summation formula j starts from 2 because at blocking state $(1, \mathrm{B})$ the transaction owns no locks and thus it cannot be involved in a deadlock.

Note that, to simplify our calculations we are considering only the occurrence of deadlocks of cycle length two (i.e., two transactions are involved in each deadlock). [9] To resolve a deadlock we abort the transaction that has just made the lock request leading to the deadlock.

The numerical solution provided in Section 4.1.3 can be used in solving the blocking and abort probabilities together with the equations for the steady-state distribution.

Sample performance experiments:

Comparison of the performances of protocols 2PL and HP was provided in a number of earlier works [1,12,28] that are all based on simulation. Evaluating both protocols under various conditions, all those works agreed that HP provides better performance than 2PL in terms of the fraction of satisfied deadlines. They also indicated that the gap between the performances of the protocols becomes larger as the level of transaction load in the system or data contention among transactions increases.

[9] It was shown elsewhere [7] that the occurrence of deadlocks of cycle length greater than two is very unlikely and can be ignored in analyzing concurrency control protocols.

In this section, we present the comparative performance results of protocols 2PL and HP obtained using the proposed analytic solution. To be able to compare our results against others', the protocols' performances were evaluated under different levels of transaction load and data contention. For that purpose, parameter t (i.e., the level of multiprogramming), D (i.e., number of data items stored in the database), and d (i.e., average number of data items accessed by each transaction) were employed in the performance experiments.

In the first experiment, the value of parameter t was varied from 5 to 45 in steps of 5, and for each value the transaction completion (deadline satisfaction) rate was calculated. The results obtained with both protocols are presented in Fig. 6. One can see that, involving real-time priorities of transactions in scheduling (i.e., using protocol HP) can provide a considerable performance improvement over 2PL. This result is due to the large blocking delays experienced by high priority transactions with 2PL. The probability of blocking (P_b) and the delay at each blocking state ($1/\mu_B$) was found to be much higher with 2PL compared to that with HP. The results also show that the difference between the performances of protocols HP and 2PL becomes much more pronounced under high transaction load conditions.

Data contention exists due to the conflicting data access requests of transactions, which results in either transaction blocking or transaction abort to resolve the conflict. In the second experiment, we studied the effects of data conflicts, and thus data contention on the comparative real-time performance of the protocols. The value of the database size D was varied from 500 to 3000 data items. As the database size increases, less data contention (due to fewer data access conflicts) occurs among concurrently executing transactions. As

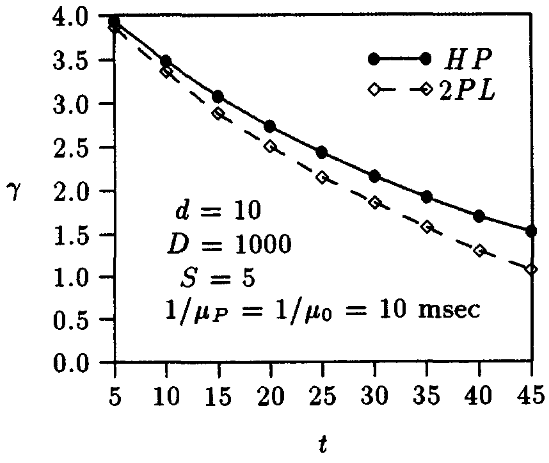

Fig. 6. Deadline satisfaction rate as a function of the multiprogramming level for protocols HP and 2PL.

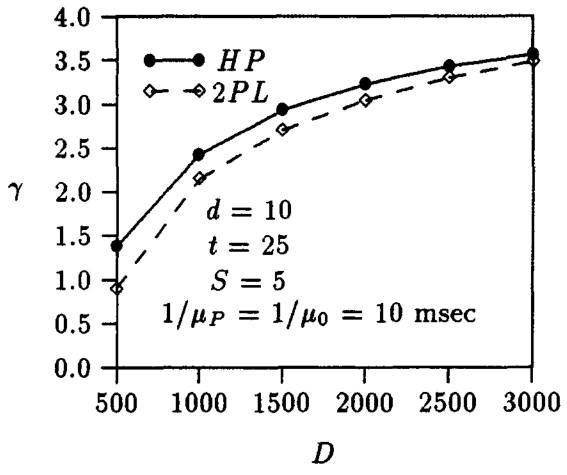

Fig. 7. Deadline satisfaction rate as a function of the database size for protocols HP and 2PL.

displayed in Fig. 7, the real-time performance improvement provided by HP over 2PL is at a higher level when the size of the shared database is small (i.e., under high levels of data contention). Another parameter we used to vary data contention was the average transaction size *d*. The range of *d* values employed in computing deadline satisfaction rate of protocols HP and 2PL was [7,13 data items]. Longer transaction lifetime results in more data conflicts and worse performance for both protocols. However, as can be seen in Fig. 8, employing protocol HP reduces the steep degradation in real-time performance which is experienced as the number of data items accessed by each transaction increases.

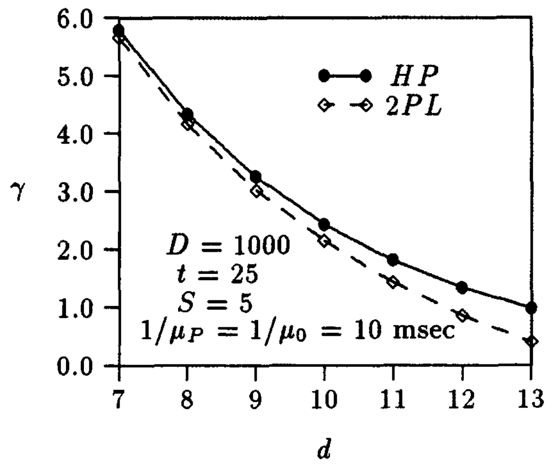

Fig. 8. Deadline satisfaction rate as a function of the transaction size for protocols HP and 2PL.

The similarity between performance results obtained by using the analytic model and those obtained previously in some simulation works (e.g., [1,12,28]) indicates that the proposed analytic solution can be considered to be a valid and useful method to predict the performance of concurrency control protocols for RTDBSs.

5. Extensions to the analysis

5.1. Considering variable size transactions

One of the assumptions of our analysis is that each transaction accesses a fixed number of data items. This section discusses how the constant size transaction assumption can be relaxed in the analysis. In modeling a variable size transaction, we adapt the following method presented in [20]: after processing a data item, a transaction commits with probability p_c or accesses another data item with probability $1 - p_c$. The number of data items that can be accessed by a transaction is bounded [10] by parameter d.

In the state-transition diagram of a transaction (Fig. 1), each processing state (i, P) with $i = 1, 2, \ldots, d-1$ has its three outward transitions updated as follows: $(i, \mathrm{P}) \rightarrow (0)$ with rate $(p_c + (1 - p_c)\mathrm{P}_{a(i,\mathrm{P})})\mu_\mathrm{P}$ (the transaction goes to state (0) whether it is committed or aborted), $(i, \mathrm{P}) \rightarrow (i+1, \mathrm{B})$ with rate $(1 - p_c)(1 - P_{a(i,\mathrm{P})})P_b\mu_\mathrm{P}$, and $(i, \mathrm{P}) \rightarrow (i+1, \mathrm{P})$ with rate $(1 - P_c)(1 - P_{a(i,\mathrm{P})})(1 - P_b)\mu_\mathrm{P}$. The steady-state probabilities should be recalculated based on the new transition values.

The completion (commit) rate of a transaction can be specified as

$$\gamma = \sum_{i=1}^{d-1} \mathbf{P}(i, \mathrm{P}) p_c \mu_\mathrm{P} + \mathbf{P}(d, \mathrm{P})\mu_\mathrm{P}.$$

Another formula affected is $\mathrm{RT}_{(i,X)}$ which represents the average remaining time of a transaction at state (i, X). Eq. (4) can be reformulated as follows:

$$\mathrm{RT}_{(i,X)} = \sum_{(j,Y)=(i,X)}^{(d,\mathrm{P})} \left(P_{c(j,Y)|(i,X)} D_{(i,X);(j,Y)}\right) + \sum_{(j,Y)=(i,X)}^{(d,\mathrm{P})} \left(P_{a(j,Y)|(i,X)} D_{(i,X);(j,Y)}\right),$$

where $P_{c(j,Y)|(i,X)}$ is the probability of commit at the end of state (j, Y) given that the current state is (i, X). In formulating $P_{c(j,Y)|(i,X)}$, X can take any one of the values B and P, while Y can take only value P since a transaction can only be committed following a processing state. If $X = \mathrm{B}$

[10] A possible variation can be to bound the transaction size by the number of data items in the database (i.e., D). In that case, the total number of states in the state transition diagram of a transaction would be $2D + 1$.

$$P_{c(j,Y)|(i,X)} = (1 - P_{a(i,B)})\prod_{k=i}^{j-1}[(1 - p_c)(1 - P_{a(k,P)})(1 - P_b P_{a(k+1,B)})]p_c.$$

Otherwise (i.e., $X = \mathrm{P}$)

$$P_{c(j,Y)|(i,X)} = \prod_{k=i}^{j-1}[(1 - p_c)(1 - P_{a(k,P)})(1 - P_b P_{a(k+1,B)})]p_c.$$

In both cases, if $j = d$, the last term of the formula (i.e., p_c) should be replaced by 1 (i.e., following the process of dth data item the transaction always commits). $P_{a(j,Y)|(i,X)}$ and $D_{(i,X);(j,Y)}$ were already calculated in Section 3.2 (see Tables 4 and 5). No further changes are required for the extension of the analysis to the case of variable transaction size.

5.2. Considering non-uniform data accesses

Our analysis has assumed that data items accessed by each transaction are uniformly distributed over the database. In this section, to allow locality to be modeled, some portion of the database is considered to be 'hot'; i.e., it is accessed more frequently than the other parts of the database. We adapt the h/p_h data access model [16], where h specifies the fraction of the hot region of the database, and p_h specifies the probability of accessing the hot region. In other words, $100 * p_h\%$ of data accesses are directed to the hot region and the remaining accesses go elsewhere (that can be called the 'cold' region) in the database. Within the hot (or cold) region, data items are chosen using a uniform distribution. The blocking and abort probabilities for protocol HP can be recalculated as follows.

5.2.1. Computation of blocking probability

$$P_b = P_{b|h}p_h + P_{b|\bar{h}}(1 - p_h),$$

$P_{b|h}\{P_{b|\bar{h}}\}$: The probability of blocking on a data access attempt given that the access is to the hot {cold} region of the database. It can be expressed in terms of Locks_hp; i.e., the average number of locks held by higher priority transactions (see Section 4.1.1):

$$P_{b|h} = \frac{p_h \mathrm{Locks\_hp}}{hD}, \tag{15}$$

p_hLocks_hp is the average number of locks in the hot region held by higher priority transactions. hD specifies the size of the hot region.

Similarly,

$$P_{b|h} = \frac{(1 - p_h)\mathrm{Locks\_hp}}{(1 - h)D}. \tag{16}$$

Substitution of Eqs. (7), (15) and (16) yields

$$P_b = \frac{((1-h)p_h^2 + h(1-p_h)^2)(t-1)L}{2h(1-h)D}.$$

5.2.2. Computation of abort probabilities

The abort probabilities due to data conflicts (i.e., $P_{a(i,B)})(1)$ and $P_{a(i,P)}(1))$ need to be reformulated:

$$P_{a(i,B)}(1) = P_{a(i,B)}(1)_{|h} p_h + P_{a(i,B)}(1)_{|\bar{h}}(1-p_h),$$

$P_{a(i,B)}(1)_{|h}\{P_{a(i,B)}(1)_{|\bar{h}}\}$: The probability that one of the locks held by the transaction is requested by a higher priority transaction, given that the requested lock is in the hot {cold} region.

Based on the calculations of Section 4.1.2,

$$\begin{aligned} P_{a(i,B)}(1)_{|h} &= \frac{p_h(i-1)}{hD} \frac{(t-1)}{2\mu_B(P_b(1/\mu_B) + 1/\mu_P)} \\ &= \frac{\mu_P}{P_b\mu_P + \mu_B} \frac{p_h(i-1)(t-1)}{2hD}, \end{aligned}$$

$$\begin{aligned} P_{a(i,B)}(1)_{|\bar{h}} &= \frac{(1-p_h)(i-1)}{(1-h)D} \frac{(t-1)}{2\mu_B(P_b(1/\mu_B) + 1/\mu_P)} \\ &= \frac{\mu_P}{P_b\mu_P + \mu_B} \frac{1-p_h(i-1)(t-1)}{2(1-h)D}. \end{aligned}$$

$P_{a(i,B)}(1)$ can then be expressed as

$$\begin{aligned} P_{a(i,B)}(1) &= p_h \frac{\mu_P}{P_b\mu_P + \mu_B} \frac{p_h(i-1)(t-1)}{2hD} \\ &\quad + (1-p_h) \frac{\mu_P}{P_b\mu_P + \mu_B} \frac{(1-p_h)(i-1)(t-1)}{2(1-h)D}. \end{aligned}$$

$P_{a(i,P)}(1)$ can be computed similarly,

$$P_{a(i,P)}(1) = p_h \frac{\mu_B}{P_b\mu_P + \mu_B} \frac{p_h i(t-1)}{2hD} + (1-p_h) \frac{\mu_B}{P_b\mu_P + \mu_B} \frac{(1-p_h)i(t-1)}{2(1-h)D}.$$

5.3. Considering shared and exclusive accesses together

So far in the analysis we have assumed that all data accesses are exclusive. This section extends the analysis by incorporating both shared and exclusive accesses. Denoting the probability of shared access by p_s, the blocking and abort probabilities for protocol HP are reformulated as follows.

5.3.1. Computation of blocking probability

$$P_b = P_{b|s}p_s + P_{b|e}(1 - p_s),$$

$P_{b|s}\{P_{b|e}\}$: The probability that the transaction is blocked given that the type of access is shared {exclusive}.

$$P_{b|s} = \frac{\text{ExLocks\_hp}}{D},$$

ExLocks_hp specifies the average number of exclusive locks currently held by higher priority transactions.

$$P_{b|e} = \frac{\text{Locks\_hp}}{D},$$

Locks_hp is the average number of locks (both shared and exclusive) held by higher priority transactions. Section 4.1.1. formulates Locks_hp in terms of the system parameters (Eq. (7)). Computation of ExLocks_hp in the same way yields

$$\text{ExLocks\_hp} = \frac{(t-1)(1-p_s)L}{2D}.$$

P_b can then be estimated as

$$P_b = p_s(1-p_s)\frac{(t-1)L}{2D} + (1-p_s)\frac{(t-1)L}{2D},$$

$$P_b = (1-p_s^2)\frac{(t-1)L}{2D}.$$

5.3.2. Computation of abort probabilities

Consideration of shared as well as exclusive accesses affects the probability of conflict aborts at blocking states (i.e., $P_{a(i,B)}(1)$) and processing states (i.e., $P_{a(i,P)}(1)$). In determining those probabilities, the computation method presented in Section 4.1.2 can be followed.

A conflict abort at a blocked state (i, B) occurs when one of the locks held by the transaction is requested by a higher priority transaction.

$$P_{a(i,B)}(1) = P_{a(i,B)}(1)_{|s}p_s + P_{a(i,B)}(1)_{|e}(1-p_s),$$

$P_{a(i,B)}(1)_{|s}\{P_{a(i,B)}(1)_{|e}\}$: The probability of conflict abort at blocking state (i, B), given that the lock requested by a high priority transaction is of type shared {exclusive}.

Based on the calculations of Section 4.1.2,

$$P_{a(i,B)}(1)_{|s} = \frac{(1-p_s)(i-1)}{D}\frac{(t-1)}{2\mu_B(P_b(1/\mu_B)+1/\mu_P)}$$
$$= \frac{\mu_P}{P_b\mu_P+\mu_B}\frac{(1-p_s)(i-1)(t-1)}{2D}.$$

Note that, $(1-p_s)(i-1)$ gives the average number of exclusive locks held by the isolated transaction at state (i, B). In calculating $P_{\mathrm{a}(i,\mathrm{B})}(1)_{|\mathrm{e}}$, on the other hand, since it is given that the lock requested by higher priority transaction is exclusive, all $(i-1)$ locks (shared or exclusive) of the isolated transaction should be considered:

$$P_{\mathrm{a}(i,\mathrm{B})}(1)_{|\mathrm{e}} = \frac{(i-1)}{D} \frac{(t-1)}{2\mu_\mathrm{B}(P_\mathrm{b}(1/\mu_\mathrm{B}) + 1/\mu_\mathrm{P})}$$
$$= \frac{\mu_\mathrm{P}}{P_\mathrm{b}\mu_\mathrm{P} + \mu_\mathrm{B}} \frac{(i-1)(t-1)}{2D}.$$

After the substitutions, $P_{\mathrm{a}(i,\mathrm{B})}(1)$ can be expressed in its final form:

$$P_{\mathrm{a}(i,\mathrm{B})}(1) = \frac{\mu_\mathrm{P}}{P_\mathrm{b}\mu_\mathrm{P} + \mu_\mathrm{B}} \frac{(1-p_\mathrm{s}^2)(i-1)(t-1)}{2D}.$$

Similarly, the computation of $P_{\mathrm{a}(i,\mathrm{P})}(1)$ (i.e., the probability that the transaction aborts at processing state (i, P) due to a data conflict) considering both shared and exclusive locks yields

$$P_{\mathrm{a}(i,\mathrm{P})}(1) = \frac{\mu_\mathrm{B}}{P_\mathrm{b}\mu_\mathrm{P} + \mu_\mathrm{B}} \frac{(1-p_\mathrm{s}^2)i(t-1)}{2D}.$$

5.4. Considering a disk-resident database

The assumption that the database is resident in main memory can be relaxed in the following way. Suppose that a data access might involve an access to disk for reading the data or for writing computed results into the database. Here we assume that these requests are served by the disk at rate μ_D according to Poisson distribution.

One method of considering both the CPU and disk access overheads is aggregating the CPU and the disk into a single load-dependent server which serves access to data items at the rate $\mu(i)$ when i transactions are being processed [21]. Assuming a processor sharing discipline, the service rate of a transaction is $\mu(i)/i$. Since the number of transactions in our system remains fixed at t, the term $\mu(t)/t$ can be replaced by a constant μ (i.e., $\mu = \mu(t)/t$). Derivation of $\mu(i)$ for queuing networks is provided using Norton's theorem in [4]. Applying that formula to our system, we obtain

$$\mu(t) = \mu_\mathrm{CPU}\left(1 - \frac{1}{1 + (\mu_\mathrm{D}/\mu_\mathrm{P}) + (\mu_\mathrm{D}/\mu_\mathrm{P})^2 + \cdots + (\mu_\mathrm{D}/\mu_\mathrm{P})^t}\right),$$

where μ_CPU is the service rate of the CPU ($\mu_\mathrm{CPU} = t\mu_\mathrm{P}$ in our system). To consider the impact of disk accesses in the analysis, μ_P (i.e., the CPU service rate per transaction) in all formulas provided in preceding sections must be replaced by the aggregate service rate μ which is equal to $\mu(t)/t$.

6. Summary and future work

This paper provided an approximate analytic solution method for evaluating the performance of priority-based concurrency control protocols developed for real-time database systems (RTDBSs). Each transaction processed in the RTDBS model employed in the evaluation was assumed to carry a priority based on its timing constraint (i.e., deadline). As a case study, the performance of a RTDBS concurrency control protocol, called High Priority (HP), was evaluated using the proposed solution method. Protocol HP is based on the two-phase locking method and it aborts a low priority transaction when one of its locks is requested by a higher priority transaction. The evaluation of HP was provided in terms of the rate of satisfying a transaction deadline. Validation of the accuracy of the results obtained by the proposed analytic solution method was performed against simulation. Results of some sample performance experiments, each evaluating the effects of a different system parameter, were presented in the paper.

The solution method was also used to model Two-Phase Locking (2PL) protocol to be able to compare the performances of protocols 2PL and HP (i.e., to evaluate the performance impact of involving real-time priorities of transactions in scheduling decisions). The performances of two protocols were compared in terms of the rate of satisfied transaction deadlines to see whether our method is capable of producing reasonable results. It was found that HP outperforms 2PL especially under high levels of transaction load. This was an obvious result confirming what was obtained in some earlier simulation works.

Several opportunities exist for expanding on the work performed. Our analysis involved a closed queuing model to keep the transaction population constant. To relax the assumption of constant transaction population the analysis can be extended to an open system which is driven by an external transaction source at a certain arrival rate and the service rate of transactions in the system is load-dependent. This appears to be a promising area for future research. Another possible extension, we are planning to work on, is considering different transaction classes in the model each having a different 'criticalness'. [11] In such RTDBS environments, the priority of a transaction is a function of both its deadline and criticalness. This extension will make it possible to evaluate the real-time performance for each class of transactions. Finally, we are also planning to apply the proposed analytic solution method to the evaluation of other concurrency control protocols developed for

[11] The criticalness of a transaction is an indication of its level of importance [3].

RTDBSs. We believe that the method can serve as a simple and fast performance evaluation tool to be used in the design and analysis of priority-based concurrency control protocols.

References

[1] R. Abbott, H. Garcia-Molina, Scheduling real-time transactions: a performance evaluation, in: Proceedings of the Fourteenth International Conference on Very Large Data Bases, 1988, pp. 1–12.

[2] R. Agrawal, M.J. Carey, M. Livny, Concurrency control performance modeling: alternatives and implications, ACM Trans. Database Systems 12 (1987) 609–654.

[3] S.R. Biyabani, J.A. Stankovic, K. Ramamritham, The integration of deadline and criticalness in hard real-time scheduling, in: Proceedings of the Ninth Real-Time Systems Symposium, 1988, pp. 152–160.

[4] K.M. Chandy, U. Herzog, L. Wu, Parametric analysis of queuing networks, IBM J. Res. Develop. 19 (1975) 36–42.

[5] A. Chesnais, E. Gelenbe, I. Mitrani, On the modeling of parallel access to shared data, Commun. ACM 26 (1983) 196–202.

[6] B. Ciciani, D.M. Dias, P.S. Yu, Performance comparison of concurrency control protocols for transaction processing systems with regional locality, in: Proceedings of the Eighth Symposium on Reliable Distributed Systems, 1989, pp. 112–118.

[7] C. Devor, C.R. Carlson, Structural locking mechanisms and their effect on database management system performance, Inform. Systems 7 (1982) 345–358.

[8] L. Gruenwald, S. Liu, A performance study of concurrency control in a real-time main memory database, ACM SIGMOD Record 22 (4) (1993) 38–44.

[9] U. Halici, A. Doğaç, An optimistic locking technique for concurrency control in distributed databases, IEEE Trans. Software Eng. 17 (7) (1991) 712–724.

[10] J.R. Haritsa, M.J. Carey, M. Livny, Dynamic real-time optimistic concurrency control, in: Proceedings of the Eleventh Real-Time Systems Symposium, 1990, pp. 94–103.

[11] J.R. Haritsa, Approximate analysis of real-time database systems, in: Proceedings of the Tenth International Conference on Data Engineering, 1994, pp. 10–19.

[12] J. Huang, J.A. Stankovic, K. Ramamritham, Experimental evaluation of real-time transaction processing, in: Proceedings of the Tenth Real-Time Systems Symposium, 1989, pp. 144–153.

[13] J. Huang, J.A. Stankovic, K. Ramamritham, D. Towsley, B. Purimetla, Priority inheritance in soft real-time databases, J. Real-Time Systems 4 (1992) 243–268.

[14] K.W. Lam, K.Y. Lam, S.L. Hung, An efficient real-time optimistic concurrency protocol, in: First Workshop on Active and Real-Time Database Systems, 1995, pp. 209–225.

[15] K.Y. Lam, S.L. Hung, Concurrency control for time-constrained transactions in distributed database systems, Comput. J. 38 (9) (1995) 704–716.

[16] W. Lin, J. Nolte, Performance of two-phase locking, in: Proceedings of the Sixth Berkeley Workshop on Distributed Data Management and Computer Networks, 1982, pp. 131–160.

[17] G. Özsoyoğlu, R.T. Snodgrass, Temporal and real-time databases: a survey, IEEE Trans. Knowledge Data Eng. 7 (1995) 513–532.

[18] M.T. Özsu, Performance comparison of distributed vs. centralized locking algorithms in distributed database systems, in: Proceedings of the Fifth International Conference on Distributed Computing Systems, 1985, pp. 254–261.

[19] L. Sha, R. Rajkumar, S.H. Son, C.H. Chang, A real-time locking protocol, IEEE Trans. Comput. 40 (1991) 793–800.

[20] M. Singhal, Performance analysis of the basic timestamp ordering algorithm via Markov modeling, Performance Evaluation 12 (1991) 17–41.
[21] M. Singhal, Analysis of the probability of transaction abort and throughput of two timestamp ordering algorithms for database systems, IEEE Trans. Knowledge Data Eng. 3 (1991) 261–266.
[22] S.H. Son, S. Park, Y. Lin, An integrated real-time locking protocol, in: Proceedings of the Eighth International Conference on Data Engineering, 1992, pp. 527–534.
[23] Y.C. Tay, N. Goodman, R. Suri, Locking performance in centralized databases, ACM Trans. Database Systems 10 (1985) 415–462.
[24] Y.C. Tay, Some performance issues for transactions with firm deadlines, in: Proceedings of the Sixteenth Real-Time Systems Symposium, 1995, pp. 322–331.
[25] A. Thomasian, I.K. Ryu, A decomposition solution to the queuing network model of the centralized DBMS with static locking, in: ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, 1983.
[26] A. Thomasian, Performance limits of two-phase locking, in: Proceedings of the Seventh International Conference on Data Engineering, 1991, pp. 426–435.
[27] Ö. Ulusoy, G.G. Belford, Real-time lock based concurrency control in a distributed database system, in: Proceedings of the Twelfth International Conference on Distributed Computing Systems, 1992, pp. 136–143.
[28] Ö. Ulusoy, G.G. Belford, Real-time transaction scheduling in database systems, Inform. System 18 (8) (1993) 559–580.
[29] Ö. Ulusoy, An approximate analysis of a real-time database concurrency control protocol via Markov modeling, Performance Evaluation Rev. 20 (3) (1993) 36–48.
[30] Ö. Ulusoy, Research issues in real-time database systems, Inform. Sci. 87 (1-3) (1995) 123–151.