An Approximate Analysis of a Real-Time Database Concurrency Control Protocol via Markov Modeling

Özgür Ulusoy
Department of Computer Engineering and Information Sciences
Bilkent University
06533 Bilkent, Ankara, TURKEY

1. Introduction

Transactions processed in a real-time database system (RTDBS) are associated with real-time constraints typically in the form of deadlines. Computer-integrated manufacturing, the stock market, banking, and command and control systems are several examples of RTDBS applications where the timeliness of transaction response is as important as the consistency of data. Design of a RTDBS requires the integration of concepts from both real-time systems and database systems to handle the timing and consistency requirements together; i.e., to execute transactions so as to both meet the deadlines and maintain the database consistency.

The general approach to the scheduling problem in RTDBS's is using 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. The timing constraints of the RTDBS transactions are considered to be 'soft'; i.e., schedules guaranteeing all transaction deadlines cannot be provided due to the consistency requirement of the underlying database. The performance goal in RTDBS scheduling is to minimize the number of transactions that miss their deadlines. A priority order is established among transactions based on their deadlines.

Most of the recent research in RTDBS transaction scheduling has concentrated on development and evaluation of concurrency control protocols. 'Priority Abort' is one of the most popular RTDB concurrency control protocols proposed so far [Abb88]. The Priority Abort protocol 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.

Performance studies concerning RTDB concurrency control protocols have appeared in the literature during the last couple of years. However, these studies were either based on simulation [Abb88, Abb89, Har90a, Har90b, Sha91, Son90, Son92, and Ulu92], or carried out on a RTDBS testbed [Hua91a, Hua91b]. To the best of our knowledge, no analytic performance study has been reported involving the evaluation of concurrency control protocols in RTDBS's.

In this paper, we analyze the performance of a RTDB concurrency control protocol, namely the Priority Abort protocol, via Markov modeling. The complexity of the concurrency control protocol makes it practically impossible to find an exact analytic closed-form solution to the performance evaluation problem. To simplify the problem, we will analyze an isolated individual transaction, rather than capturing the states of all concurrent transactions. This method, proposed by [Che83], was found to be fairly accurate in analyzing the performance of two-phase locking [Che83] and timestamp-ordering algorithms [Sin91]. 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 the transactions is modeled rather than their dynamic behavior.

The remainder of the paper is organized as follows. The next section describes the priority-based

```
lock_request_handling(D,T) {
    /* Transaction T requests a lock on data item D */
    if (D was locked by a transaction T')
        if (priority(T) > priority(T')) {
            T' is aborted;
            Lock on D is granted to T;
        }
        otherwise
            T is blocked by T';
    otherwise
        Lock on D is granted to T;
}
```

Figure 1: Lock request handling in Priority Abort protocol.

concurrency control protocol we studied. Section 3 provides the structure and characteristics of a RTDBS model used in the evaluation of the concurrency control protocol. The analysis of the performance of the protocol is presented in Section 4. Section 5 provides the results of some experiments performed by using the proposed analytic model. Finally, Section 6 provides a brief discussion of our work together with the future plans.

2. Description of the Priority Abort Protocol

In this protocol, the winner in the case of a lock conflict between two transactions is always the higher priority transaction [Abb88]. 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 (Figure 1).

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.

3. RTDBS Model

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

Each transaction submitted to the system is associated with a deadline, and is assigned a unique real-time priority determined on the basis of its deadline. 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 a factor of the estimated response time of the transaction, and that factor is determined by the parameter S.

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 the transactions are controlled by using the Priority Abort protocol. Depending on its real-time priority, an access

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
μ_P	Mean CPU service rate (transaction/msec)
S	Slack factor used in assigning transaction deadlines

Table 1: Parameters of the RTDBS Model

request of a transaction is either granted or results in blocking 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 holds after it has either been committed or aborted. A transaction can be committed after it has processed the last data item in its access list. An executing transaction can be aborted due to any of the two basic reasons: whenever its deadline expires or one of its locks is requested by a higher priority transaction.

The other primary assumptions adopted in our model simplifying the analysis are:

- The transaction population in the system (the level of multiprogramming) is constant and determined by the parameter t.

- Each transaction accesses the same number of data items, which is specified by the parameter d.

- Data items accessed by each transaction are uniformly distributed over all database.

- 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.

Table 1 summarizes the parameters of the RTDBS model.

4. Performance Analysis of the Priority Abort Protocol

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 Figure 2. State (0) of the chain represents the initialization phase of the transaction. It is assumed that the initialization phase is distributed exponentially with mean $1/\mu_0$. The other $2d$ states are labeled by a tuple (i,X), where i is an integer which can take any value from the set $\{1, 2, ..., d\}$, and denotes that the transaction is accessing its ith data item. X can take either of the two values: B or P. The access request of the transaction on a data item results in either blocking of the transaction (with probability P_b), or allowing it to acquire the lock on the requested item (with probability $1 - P_b$). 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 $i \in \{1,2,...,d\}$. Sections 4.1 and 4.2 provide the methods used in deriving P_b and μ_B, respectively. State (i,P), denotes the case that
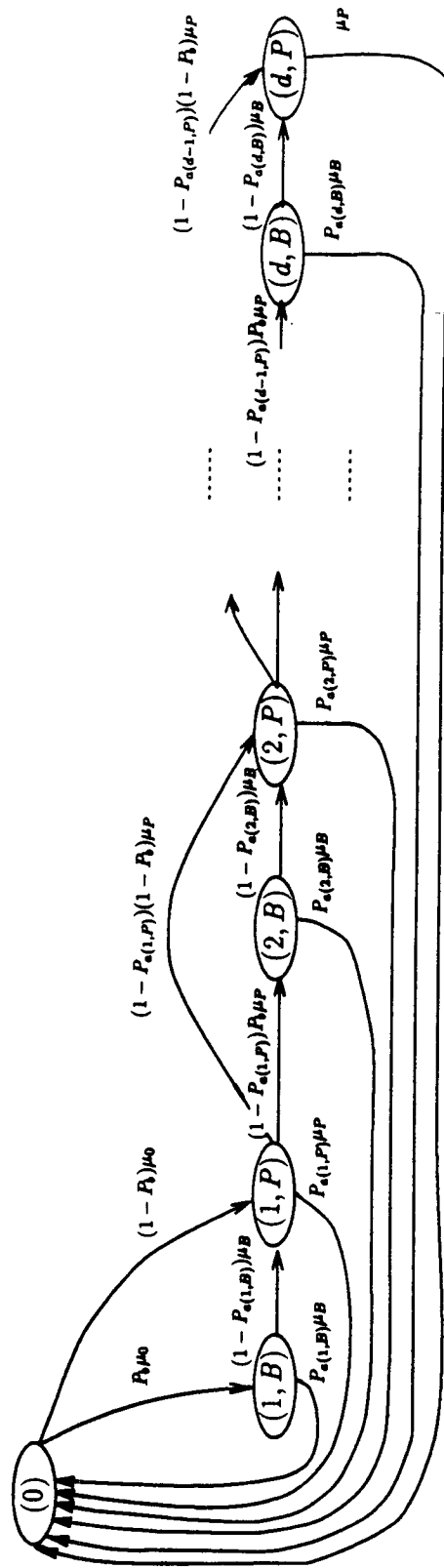
Figure 2: State-transition diagram for a transaction.

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 $(2 \leq i \leq d)$ is performed before leaving the processing state $(i - 1, P)$. At any state (i, X), it is possible that the transaction is 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 $P_{a(i,B)}$ and $P_{a(i,P)}$, 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., $P_{a(d,P)} = 0$.

Let $\{P(0), P(1, B), P(1, P), P(2, B), P(2, P), ..., P(d, B), P(d, P)\}$ be the steady-state distribution of the Markov chain. The following system of linear equations can be given for this distribution:

$$P(1, B) = \frac{\mu_0}{\mu_B} P_b P(0)$$

$$P(1, P) = \frac{\mu_0}{\mu_P}(1 - P_b P_{a(1,B)}) P(0)$$

$$P(2, B) = \frac{\mu_0}{\mu_B} P_b (1 - P_{a(1,P)})(1 - P_b P_{a(1,B)}) P(0)$$

$$P(2, P) = \frac{\mu_0}{\mu_P}(1 - P_b P_{a(1,B)})(1 - P_{a(1,P)})(1 - P_b P_{a(2,B)}) P(0)$$

$$P(3, B) = \frac{\mu_0}{\mu_B} P_b (1 - P_{a(1,P)})(1 - P_b P_{a(1,B)})(1 - P_{a(2,P)})(1 - P_b P_{a(2,B)}) P(0)$$

$$P(3, P) = \frac{\mu_0}{\mu_P}(1 - P_b P_{a(1,B)})(1 - P_{a(1,P)})(1 - P_b P_{a(2,B)})(1 - P_{a(2,P)})(1 - P_b P_{a(3,B)}) P(0)$$

$$\vdots$$

$$P(i, B) = \frac{\mu_0}{\mu_B} P_b \prod_{k=1}^{i-1}[(1 - P_{a(k,P)})(1 - P_b P_{a(k,B)})] P(0) \qquad i \in \{1, 2, ..., d\} \qquad (1)$$

$$P(i, P) = \frac{\mu_0}{\mu_P}(1 - P_b P_{a(1,B)}) \prod_{k=1}^{i-1}[(1 - P_{a(k,P)})(1 - P_b P_{a(k+1,B)})] P(0) \qquad i \in \{1, 2, ..., d\} \quad (2)$$

$$P(0) + \sum_{i=1}^{d}(P(i, B) + P(i, P)) = 1 \qquad (3)$$

The system can be solved by first determining $P(0)$ by substituting Equations 1 and 2 in Equation 3, and then computing the other steady-state probabilities $P(i, B)$, $P(i, P)$ $(1 \leq i \leq d)$ from

Equations[1] 1 and 2. However, the solution to each of these probabilities is provided in terms of P_b, μ_B, $P_{a(i,B)}$, and $P_{a(i,P)}$. Computation of each of those variables is provided in the following subsections.

4.1. Computation of P_b

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:

$$P_b = \frac{Locks\_HP}{D}$$

$Locks\_HP$ 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, ..., $(t - 1)$ with equal probability. That is, the average number of transactions with higher priorities will be $(0 + 1 + 2 + ... + (t - 1))/t = (t - 1)/2$. Let L denote the average number of data items locked by the isolated transaction. L can be formulated as a function of the steady-state distribution.

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

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

$$Locks\_HP = \frac{(t - 1)L}{2}$$

P_b can then be expressed as

$$P_b = \frac{(t - 1)L}{2D} \tag{4}$$

4.2. Computation of μ_B

When a transaction T is blocked by another transaction T' on a data item, transaction T is not reactivated until after transaction 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[2] the current state of T. 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 $RT_{(iX)}$ of the transaction can be determined by the following formula

$$RT_{(iX)} = P_{COMMIT|(i,X)}D_{(i,X);COMMIT} + \sum_{(j,Y)=(i,X)}^{(d,P)} (P_{a(j,Y)|(i,X)}D_{(i,X);(j,Y)})$$

---

[1]Note that, the equations assume $\prod_{i=a}^{b} f(i) = 1$, $if\ a < b$.

[2]Assuming that the number of locks held by the transaction $\ll D$ (database size).

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

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

X	$D_{(i,X);COMMIT}$
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 3: Average distance $(D_{(i,X);COMMIT})$ from state (i,X) to commit.

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

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

X	Y	$D_{(i,X);(j,Y)}$
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	$(j - i)(P_b\frac{1}{\mu_B} + \frac{1}{\mu_P})$

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

where, $P_{COMMIT|(i,X)}$ 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 $\prod_{i=a}^{b} f(i) = 1$, $if \; a < b$); $D_{(i,X);COMMIT}$ is the average time distance between state (i, X) and the commit time (see Table 3); $P_{a(j,Y)|(i,X)}$ 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 (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

$$\frac{1}{\mu_B} = \mathbf{P}(1,P)RT_{(1,P)} + \mathbf{P}(2,B)RT_{(2,B)} + \mathbf{P}(2,P)RT_{(2,P)} + ... + \mathbf{P}(d,B)RT_{(d,B)} + \mathbf{P}(d,P)RT_{(d,P)}$$

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

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

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 to be detailed in Section 5.

4.3. Computation of Abort Probabilities

The transaction can be aborted at any state (i, X) (where $i \in \{1, 2, ..., d\}$, and $X \in \{B, P\}$) 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.

$$P_{a(i,B)} = P_{a(i,B)}(1) + P_{a(i,B)}(2) - P_{a(i,B)}(1) * P_{a(i,B)}(2) \qquad i \in \{1, 2, ..., d\} \tag{6}$$

$$P_{a(i,P)} = P_{a(i,P)}(1) + P_{a(i,P)}(2) - P_{a(i,P)}(1) * P_{a(i,P)}(2) \qquad i \in \{1, 2, ..., d\} \tag{7}$$

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.

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

$$P_{a(i,B)}(1) = \frac{(i-1)}{D} \frac{(t-1)}{2\mu_B(P_b(1/\mu_B)+1/\mu_P)} = \frac{\mu_P}{P_b\mu_P+\mu_B} \frac{(i-1)(t-1)}{2D} \tag{8}$$

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.

$$P_{a(i,P)}(1) = \frac{\mu_B}{P_b\mu_P+\mu_B} \frac{i(t-1)}{2D} \tag{9}$$

It is assumed that D is assigned a value large enough to produce a sensible solution for the probabilities (i.e., a solution 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

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

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 $AGE_{(i,X)}$,

$$P_{a(i,B)}(2) = \frac{AGE_{(i,B)}}{ST}$$

$$P_{a(i,P)}(2) = \frac{AGE_{(i,P)}}{ST}$$

where,

$$AGE_{(i,B)} = \frac{1}{\mu_0} + (i-1)(P_b\frac{1}{\mu_B} + \frac{1}{\mu_P}) + \frac{1}{\mu_B}$$

$$AGE_{(i,P)} = \frac{1}{\mu_0} + i(P_b\frac{1}{\mu_B} + \frac{1}{\mu_P})$$

Substitution of the average age and slack time parameters yields

$$P_{a(i,B)}(2) = \frac{\frac{1}{\mu_0} + (i-1)(\frac{P_b}{\mu_B} + \frac{1}{\mu_P}) + \frac{1}{\mu_B}}{S(\frac{1}{\mu_0} + d(\frac{P_b}{\mu_B} + \frac{1}{\mu_P}))} \tag{10}$$

$$P_{a(i,P)}(2) = \frac{\frac{1}{\mu_0} + i(\frac{P_b}{\mu_B} + \frac{1}{\mu_P})}{S(\frac{1}{\mu_0} + d(\frac{P_b}{\mu_B} + \frac{1}{\mu_P}))} \tag{11}$$

Abort probabilities $P_{a(i,B)}$ and $P_{a(i,P)}$ can be expressed in their final forms by substituting Equations 8, 9, 10, and 11 in Equations 6, and 7.

4.4. Performance Metrics

We are primarily interested in the rate a transaction satisfies its deadline. The transaction completion rate would 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

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

Another performance metric that can be used in evaluations is the ratio of the transaction completion rate over the transaction start rate. Denoting this metric by *success_ratio*,

$$success\_ratio = \frac{\gamma}{\mathbf{P}(0)\mu_0}$$

This metric specifies the fraction of transactions that are able to commit successfully meeting their deadlines.

5. Numerical Solution and Results

Figure 3 presents the procedure employed in solving the linear system of equations for the steady-state distribution (i.e., Equations 1 through 3), blocking probability (i.e., Equation 4), average blocking delay (i.e., Equation 5), and aborting probabilities (i.e., Equations 6 and 7). As mentioned before, a numerical iteration is needed in computing the value of the average blocking delay $(1/\mu_B)$ because a choice for μ_B determines the steady-state probabilities which when substituted in Equation 5 generates a new computed value for μ_B.

It was observed 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 $(1/\mu_B)$ value of $d/2\mu_P$, 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 Equation 4.

In this section, we provide the results of two experiments that evaluated the performance of the Priority Abort protocol, in terms of transaction completion rate (γ), using the proposed analytic solution model. The size of the database chosen for the analysis 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 the transactions. This small database can be considered as the most frequently accessed fraction of a larger database. The average service time at the CPU for processing a data item was taken as $1/\mu_P = 10$ msec. We assumed that the average delay for transaction initialization is the same as the average CPU service time $(1/\mu_0 = 1/\mu_P)$. Calculations in both experiments were performed under three different multiprogramming levels; i.e., $t = 5$, 15, and 25 transactions.

The first experiment investigated the impact of varying average transaction size on the performance of the Priority Abort protocol. The parameter d was varied from 5 to 15 in steps of 2. The slack

```
solution_procedure {
    μ'_B = 0
    initialize P_b, μ_B
    while ( |μ_B − μ'_B| / μ_B > ε) {
        Compute P_{a(i,X)} = P_{a(i,X)}(P_b, μ_B), i ∈ {1,2,...,d}; X ∈ {B,P}
        Compute P(0) = P(0)(P_b, μ_B, P_{a(j,Y)}),
                P(i,X) = P(i,X)(P_b, μ_B, P_{a(j,Y)})
                i,j ∈ {1,2,...,d}; X,Y ∈ {B,P}
        P_b = P_b(P(0), P(i,X)), i ∈ {1,2,...,d}; X ∈ {B,P}
        μ'_B = μ_B
        μ_B = μ_B(μ'_B, P(0), P(i,X), P_b, P_{a(j,Y)}), i,j ∈ {1,2,...,d}; X,Y ∈ {B,P}
```

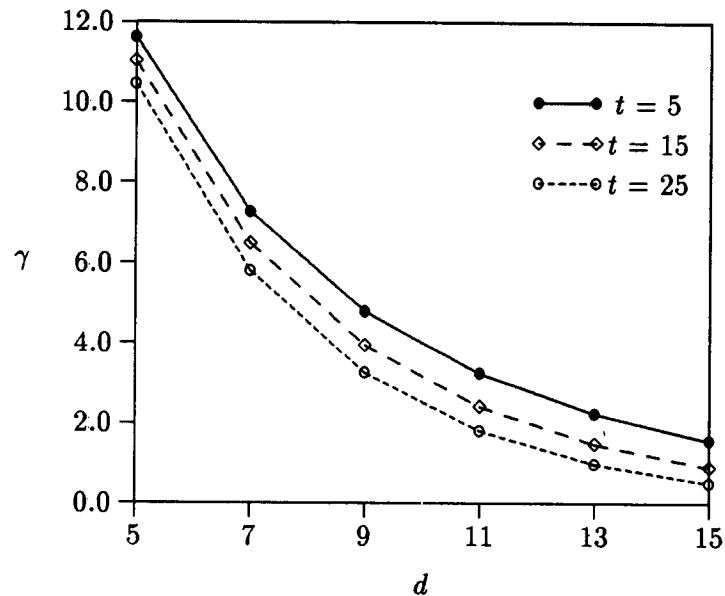Figure 3: System solution by numerical iteration.

Figure 4: Transaction completion rate (transaction/second) vs d (average number of data items accessed by each transaction).
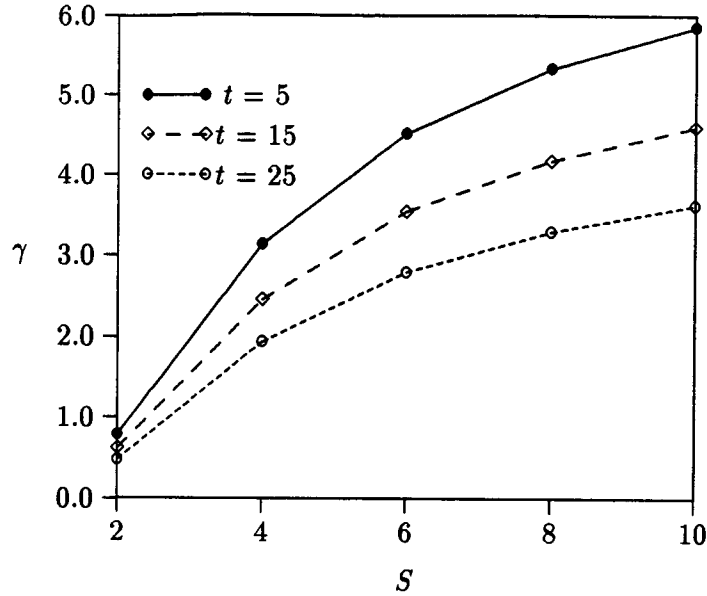
Figure 5: Transaction completion rate (transaction/second) vs S (slack factor that is used in assigning deadline to a new transaction).

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 Figure 4, the transaction completion rate (or equivalently, the deadline satisfaction rate) decreases drastically as the number of data items accessed by each transaction increases.

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 Figure 5.

6. Discussion and Future Work

This paper provides an approximate analytic model for evaluating the performance of a priority-based concurrency control protocol for real-time database systems. Each transaction processed in the system is assumed to carry a priority based on its deadline. The protocol 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 the protocol is provided in terms of the rate of satisfying a transaction deadline. The results of some example experiments, performed using the proposed analytic model, are presented in the paper.

The future work includes performing a variety of simulation experiments to evaluate the accuracy of the provided analytic model. Also, planned as a future work is the relaxation of some of the basic assumptions listed in Section 3 to extend our analysis to more general real-time database systems.

References

[Abb88]: R. Abbott, H. Garcia-Molina 'Scheduling Real-Time Transactions: A Performance Evaluation', *14th International Conference on Very Large Data Bases*, 1988, pp.1-12.

[Abb89]: R. Abbott, H. Garcia-Molina 'Scheduling Real-Time Transactions with Disk Resident Data', *15th International Conference on Very Large Data Bases*, 1989, pp.385-396.

[Che83]: A. Chesnais, E. Gelenbe, I. Mitrani 'On the Modeling of Parallel Access to Shared Data', *Communications of the ACM*, vol.26, no.3, 1983, pp.196-202.

[Har90a]: J.R. Haritsa, M.J. Carey, M. Livny 'On Being Optimistic About Real-Time Constraints', *ACM SIGACT-SIGMOD-SIGART*, 1990, pp.331-343.

[Har90b]: J.R. Haritsa, M.J. Carey, M. Livny 'Dynamic Real-Time Optimistic Concurrency Control', *11th Real-Time Systems Symposium*, 1990, pp.94-103.

[Hua91a]: J. Huang, J.A. Stankovic, K. Ramamritham, D. Towsley 'On Using Priority Inheritance In Real-Time Databases', *12th Real-Time Systems Symposium*, 1991, pp.210-221.

[Hua91b]: J. Huang, J.A. Stankovic, K. Ramamritham, D. Towsley 'Experimental Evaluation of Real-Time Optimistic Concurrency Control Schemes', *17th International Conference on Very Large Data Bases*, 1991, pp.35-46.

[Sha91]: L. Sha, R. Rajkumar, S.H. Son, C.H. Chang 'A Real-Time Locking Protocol', *IEEE Transactions on Computers*, vol.40, no.7, 1991, pp.793-800.

[Sin91]: M. Singhal 'Performance Analysis of the Basic Timestamp Ordering Algorithm via Markov Modeling', *Performance Evaluation*, vol.12, 1991, pp.17-41.

[Son90]: S.H. Son, C.H. Chang 'Performance Evaluation of Real-Time Locking Protocols Using a Distributed Software Prototyping Environment', *10th International Conference on Distributed Computing Systems*, 1990, pp.124-131.

[Son92]: S.H. Son, S. Park, Y.Lin 'An Integrated Real-Time Locking Protocol', *8th International Conference on Data Engineering*, 1992, pp.527-534.

[Ulu92]: Ö. Ulusoy, G.G. Belford 'Real-Time Lock Based Concurrency Control in a Distributed Database System', *12th International Conference on Distributed Computing Systems*, 1992, pp.136-143.