Fixed Priority Scheduling of Age Constraint Processes

Lars Lundberg

Department of Computer Science, University of Karlskrona/Ronneby,
S-372 25 Ronneby, Sweden,
Lars.Lundberg@ide.hk-r.se

Abstract. Real-time systems often consist of a number of independent processes which operate under an age constraint. In such systems, the maximum time from the start process L_i in cycle k to the end in cycle $k+1$ must not exceed the age constraint A_i for that process. The age constraint can be met by using fixed priority scheduling and periods equal to $A_i/2$. However, this approach restricts the number of process sets which are schedulable.
In this paper, we define a method for obtaining process periods other than $A_i/2$. The periods are calculated in such a way that the age constraints are met. Our approach is better in the sense that a larger number of process sets can be scheduled compared to using periods equal to $A_i/2$.

1 Introduction

Real-time systems often consist of a number of independent periodic processes. These processes may handle external activities by monitoring sensors and then producing proper outputs within certain time intervals. A similar example is a process which continuously monitors certain variables in a database. When these variables or sensors change, the system have to produce certain outputs within certain time intervals. These outputs must be calculated from input values which are fresh, i.e. the age of the input value must not exceed certain time limits. The processes in these kinds of systems operate under the age constraint.

The age constraint defines a limit on the maximum time from the point in time when a new input value appears to the point in time when the appropriate output is produced. Figure 1 shows a scenario where a value E_i appears shortly after process L_i has started its k:th cycle (denoted L_i^k). Process L_i starts its execution by reading the sensor or variable. Consequently, E_i will not affect the output in cycle k. The output F_i corresponding to value E_i (or fresher) is produced at the end of cycle $k+1$. The age constraint A_i is defined as the maximum time between the beginning of the process' execution in cycle k to the end of the process' execution in cycle $k+1$.

A scheduling scheme can be either static or dynamic. In dynamic schemes the priority of a process is decided at run-time, e.g. the earliest deadline algorithm [3]. In static schemes, processes are assigned a fixed priority, e.g. the rate-monotone algorithm [4]. Fixed priority scheduling is relatively easy to implement and it requires less overhead than dynamic schemes.

Most studies in this area have looked at scenarios where the computation time and the period of a process are known. However, for age constraint processes the period is not known. We have instead defined a maximum time between the beginning of the process' execution in cycle k to the end of the process' execution in cycle k+1. We would like to translate this restriction into a period for the process thus making it possible to use fixed priority schemes.

The age constraint is met by specifying a process period $T_i' = A_i/2$ (we will use the notation T_i for other purposes), thus obtaining a set of processes with known periods T_i' and computation times C_i. For such scenarios, it is well known that rate-monotone scheduling is optimal, and a number of schedulability tests have been obtained [4]. Specifying a process period $T_i' = A_i/2$ for age constraint processes is, however, an unnecessary strong restriction, which do not allow that the start of L_i in cycle k and the end in cycle k+2 may be separated by a time greater than $3T_i'$, whereas the age constraint allows a separation of up to $2A_i = 4T_i'$.

In this paper we show that, by using rate-monotone scheduling, it is possible to define periods which are better than using periods $T_i' = A_i/2$. Our method is better in the sense that we will be able to schedule a larger number of process sets than using $T_i' = A_i/2$.

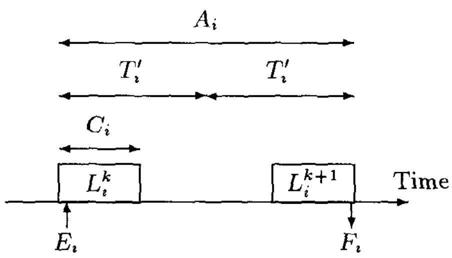

Fig. 1. The age constraint for process L_i.

2 Calculating process periods

Consider a set of n processes $\overline{L} = [L_1, L_2, ..., L_n]$, with associated age constraints A_i and computation times C_i. The priority of each process is defined by its age constraint A_i. The smaller the value A_i, the higher the priority of L_i, i.e. the priority order is the same as for rate-monotone scheduling with $T_i' = A_i/2$. We assume preemptive scheduling and we order the processes in such a way that $A_i \leq A_{i+1}$.

We start by considering L_1. This process has the highest priority, and is thus not interrupted by any other process. We want to select as long periods as possible, thus minimizing processor utilization. Obviously, the period of a process must not exceed $A_i - C_i$. Since L_1 has the highest priority, it is safe to set the period of L_1 to $A_1 - C_1$ In order to distinguish our periods from the ones which are simply equal to half the age constraint A_i, we denote our periods as T_i, i.e. $T_1 = A_1 - C_1$.

We now consider process L_2. The maximum response time R_2 of process L_2 is defined as the maximum time from the release of L_2 in cycle k to the time that L_2 completes in the same cycle. If the execution of L_2 is not interrupted by any other process, the response time is simply equal to the computation time C_2, i.e. $R_1 = C_1$. However, the execution of L_2 may be interrupted by L_1. Process L_1 may in fact interfere with as much as $\lceil R_2/T_1 \rceil C_1$ [3]. Consequently, $R_2 = C_2 + \lceil R_2/T_1 \rceil C_1$. The only unknown value in this equation is R_2. The equation is somewhat difficult to solve due to the ceiling function ($\lceil R_2/T_1 \rceil$). In general there could be many values of R_2 that solve this equation. The smallest such value represents the worst-case response time for process L_2. It has been shown that R_2 can be obtained from this equation by forming a recurrence relationship. The technique for doing this is shown in [3].

The beginning of L_2 in cycle k and the end of L_2 in cycle k+1 may be separated with as much as $T_2 + R_2$, where T_2 is the period that we will assign to process L_2. From the age constraint we know that $T_2 + R_2 \leq A_2$. In order to minimize processor utilization we would like to select as long a period T_2 as possible. Consequently, $T_2 = A_2 - R_2$.

In general, the maximum response time of process i can be obtained from the relation $R_i = C_i + \sum_{j=1}^{i-1} \lceil R_i/T_j \rceil C_j$ [3]. When we know R_i, the cycle time for L_i is set to $T_i = A_i - R_i$.

Figure 2 shows a set with three processes L_1, L_2 and L_3, defined by $A_1 = 8$, $C_1 = 2$, $A_2 = 10$, $C_2 = 2$, $A_3 = 12$ and $C_3 = 2$. From these values we obtain the period $T_1 = A_1 - C_1 = 8 - 2 = 6$. The maximum response time R_2 for process L_2 is obtained from the relation $R_2 = C_2 + \lceil R_2/T_1 \rceil C_1 = 2 + \lceil R_2/6 \rceil 2$. The smallest value R_2 which solves this equation is 4, i.e. $R_2 = 4$. Consequently, $T_2 = A_2 - R_2 = 10 - 4 = 6$. The maximum response time R_3 for process L_3 is obtained from the relation $R_3 = C_3 + \lceil R_3/T_1 \rceil C_1 + \lceil R_3/T_2 \rceil C_2 = 2 + \lceil R_3/6 \rceil 2 + \lceil R_3/6 \rceil 2$. The smallest value R_3 which solves this equation is 6, i.e. $R_3 = 6$. Consequently, $T_3 = A_3 - R_3 = 12 - 6 = 6$. In figure 2, the first release of L_1 is done at time 2, the first release of L_2 is done at time 1 and the first release of L_3 is done at time 0.

In the worst-case scenario, process L_i may suffer from the maximum response time R_i in two consecutive cycles, i.e. in order to meet the age constraint we know that $2Ri \leq A_i$. Consequently, there is no use in selecting a T_i smaller than R_i. However, as long as we obtain T_i which are longer than or equal to R_i, the age constraint will be met. Therefore, process L_i can be scheduled if and only if $R_i \leq A_i/2$.

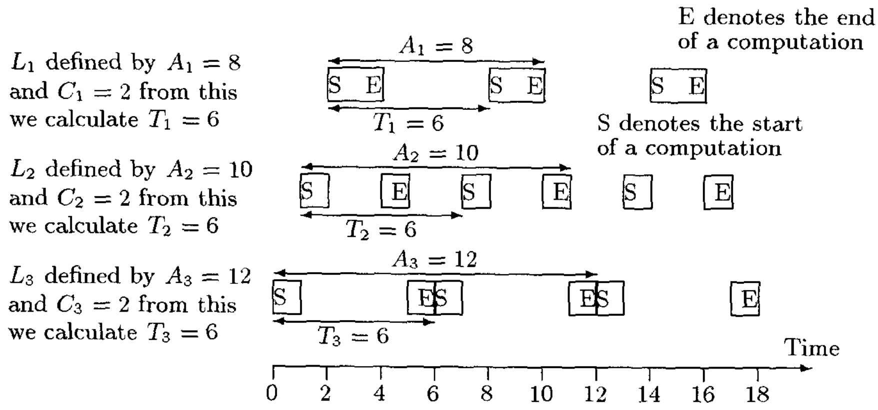

Fig. 2. A set with three processes L_1, L_2 and L_3, defined by $A_1 = 8, C_1 = 2, A_2 = 10, C_2 = 2, A_3 = 12$ and $C_3 = 2$.

Theorem 1. *A set of processes which is schedulable using rate-monotone priority assignment and $T'_i = A_i/2$ is also schedulable using our scheme.*

Proof. The difference between our scheme and rate-monotone with $T'_i = A_i/2$ is that we use different periods T_i. Since the process set is schedulable using the T'_i periods we know that $R'_i \leq T'_i (1 \leq i \leq n)$, where R'_i denotes the maximum response time using the T'_i periods.

By use of induction, we show that $R_i \leq R'_i (1 \leq i \leq n)$.

- $R_1 = C_1 \leq R'_1 = C_1$
- If $R_j \leq R'_j (1 \leq j \leq x < n)$, then $T'_j = A_j/2 \leq A_j - R_j = T_j$. If $T'_j \leq T_j$, then $R_{x+1} = C_{x+1} + \sum_{j=1}^{x} \lceil R_{x+1}/T_j \rceil C_j \leq R'_{x+1} = C_{x+1} + \sum_{j=1}^{x} \lceil R'_{x+1}/T'_j \rceil C_j$.

Consequently, $R_i \leq R'_i \leq T'_i = A_i/2$, i.e. $R_i \leq A_i/2$, which means that process $L_i (1 \leq i \leq n)$ can be scheduled using our scheme.

Consider the processes in figure 2. If we would have used the periods $A_i/2$, we would have got $T'_1 = 4, T'_2 = 5$ and $T'_3 = 6$. This would have resulted in a utilization of $C_1/T'_1 + C_2/T'_2 + C_3/T'_3 = 2/4 + 2/5 + 2/6 = 1.23 > 1$, i.e. the process set would not have been schedulable. Consequently, our scheme is better than rate-monotone and $T'_i = A_i/2$ in the sense that we are able to schedule a larger number of process sets.

3 Simple analysis

In the scheme that we propose, the period of a process depends on the priority of the process. The period for a process L_i gets shorter if the priority of L_i is reduced and vice versa. This property makes our scheme hard to analyze. However, for the limited case when there are two processes, a thorough analysis is possible.

Theorem 2. *The priority assignment used in our scheme is optimal for all sets containing two processes.*

Proof. Consider two processes L_1 and L_2, such that $A_1 \leq A_2$. Assume that these two processes are schedulable if the priority of L_2 is higher than the priority of L_1. We will now show that if this is the case, the two processes are also schedulable if L_1 has higher priority than L_2.

If L_1 and L_2 are schedulable when the priority of L_2 is higher than the priority of L_1, then $R_1 = C_1 + \lceil R_1/(A_2 - C_2) \rceil C_2 = C_1 + kC_2 \leq A_1/2$ (for some integer $k > 0$). Consequently, $C_2 \leq A_1/2 - C_1$.

If we consider the opposite priority assignment we know that the schedulability criterion is that $R_2 = C_2 + \lceil R_2/(A_1 - C_1) \rceil C_1 \leq A_2/2$. Since $C_2 \leq A_1/2 - C_1$, we know that the maximum interference from process L_1 on process L_2 is C_1, i.e. $R_2 = C_2 + C_1$. Since $C_2 \leq A_1/2 - C_1$ and $R_2 = C_2 + C_1$, we know that $R_2 \leq A_1/2$, and since $A_1 \leq A_2$, we know that $R_2 \leq A_2/2$, thus proving the theorem.

Theorem 3. *All sets containing two processes L_1 and L_2 for which $C_1/(A_1 - C_1) + C_2/(A_2 - C_2)$ is less than $2(\sqrt{2} - 1) = 0.83$ are schedulable using our scheme, and there are process sets containing two processes for which $C_1/(A_1 - C_1) + C_2/(A_2 - C_2) = 2(\sqrt{2} - 1) + e$ (for any $e > 0$) which are not schedulable using our scheme.*

Proof. We assume that $A_1 \leq A_2$, and that we use our scheme for calculating periods and priorities.

We want to find the minimum value $C_1/(A_1 - C_1) + C_2/(A_2 - C_2)$, such that the process set is not schedulable. We know that as long as $C_1/(A_1 - C_1) \leq 1$, process L_1 can be scheduled. This is a trivial observation.

Process L_2 can be scheduled if $R_2 \leq A_2/2$, i.e. we want to minimize $C_1/(A_1 - C_1) + C_2/(A_2 - C_2)$ under the constraint that $R_2 = C_2 + \lceil R_2/(A_1 - C_1) \rceil C_1 > A_2/2$.

If $\lceil R_2/(A_1 - C_1) \rceil = k$ (for some integer $k > 0$), then the minimum for $C_1/(A_1 - C_1) + C_2/(A_2 - C_2)$ is obtained when $R_2 = k(A_1 - C_1) = C_2 + \lceil R_2/(A_1 - C_1) \rceil C_1 = C_2 + kC_1 \Rightarrow C_2 = k(A_1 - 2C_1)$. Consequently, we want to find the k which minimizes $C_1/(A_1 - C_1) + k(A_1 - 2C_1)/(A_2 - k(A_1 - 2C_1))$. Since, $C_2 = k(A_1 - 2C_1) \leq A_2/2$ we see that $A_2 - k(A_1 - 2C_1) > 0$, and since $0 \leq A_1 - 2C_1$, we see that the minimum for $C_1/(A_1 - C_1) + k(A_1 - 2C_1)/(A_2 - k(A_1 - 2C_1))$ is obtained for $k = 1$. Consequently, we want to minimize $C_1/(A_1 - C_1) + (A_1 - 2C_1)/(A_2 - (A_1 - 2C_1))$, under the constraint that $R_2 = A_1 - C_1 > A_2/2$. The minimum is obviously obtained when $A_1 - C_1$ is as small as possible, i.e. when $A_1 - C_1 = A_2/2 + e$ (for some infinitely small positive number e). Consequently, we want to minimize $C_1/(A_2/2 + e) + (A_2/2 + e - C_1)/(A_2 - (A_2/2 + e - C_1))$. Without loss of generality we assume that $A_2 = 2$. In that case we obtain the following function (disregarding e)

$$f(C_1) = C_1 + (1 - C_1)/(1 + C_1)$$

From this we obtain the derivative of f:

$$f'(C_1) = 1 + ((1 - C_1) - (1 + C_1))/(1 + C_1)^2$$

By setting $f'(C_1) = 0$ we will find the values C_1 which minimizes f.

$$1+((1-C_1)-(1+C_1))/(1+C_1)^2 = 0 => 1+2C_1+C_1^2+1-2C_1 = 0 => C_1 = \sqrt{2}-1$$

From this we that min $f(C_1) = f(\sqrt{2} - 1) = 2(\sqrt{2} - 1) = 0.83$.

It is interesting to note that the schedulability bound for $C_1/(A_1 - C_1) + C_2/(A_2 - C_2)$ is the same as the schedulability bound for rate-monotone scheduling and two processors. However, in that case we have $C_1/T_1' + C_2/T_2' = 2C_1/A_1 + 2C_2/A_2 = 0.83$. At this point we do not know if it is a coincident that the values are the same or not. It would be interesting to examine the case with three processes and see if the schedulability bound for $C_1/(A_1 - C_1) + C_2/(A_2 - C_2) + C_3/(A_3 - C_3) = 3(\sqrt[3]{2} - 1) = 0.78$, which is the bound for rate-monotone scheduling and three processes.

4 Improving the scheme

In the previous sections we assumed that the interference from a higher priority process L_j affected the maximum response time R_{j+x} for a process L_{j+x} according to the formula $R_{j+x} = C_{j+x} + \cdots + \lceil R_{j+x}/T_j \rceil C_j$. However, if $T_{j+x} = kT_j$ (for some integer $k > 0$), we can adjust the phasing of L_j and L_{j+x} in such a way that the interference of L_j on L_{j+x} is limited to $(\lceil R_{j+x}/T_j \rceil - 1)C_j$ (see figure 3). The phasing is adjusted in such a way that a release of process L_{j+x} always occurs at exactly the same time as a release of process L_j. Consequently, if $T_{j+x} < kT_j \le T_{j+x} - C_j$ we can extend the period of L_{j+x} to kT_j. In order to distinguish the periods obtained when using the optimized version of the scheme from the ones obtained using the unoptimized version, we denote the optimized period for process L_j as t_j.

In order to obtain the optimized periods t_j, for a set containing n processes, we start with the periods T_j and we then use the following algorithm:

$t_1 = T_1$
for $y = 2$ to n loop
    $t_y = T_y$
    for $j = 1$ to $y - 1$ loop
        if $t_y < kt_j \le T_y - C_j$ then $t_y = kt_j$
    end loop
    $y = y + 1$
end loop

The execution of the process set is started by releasing all processes at the same time.

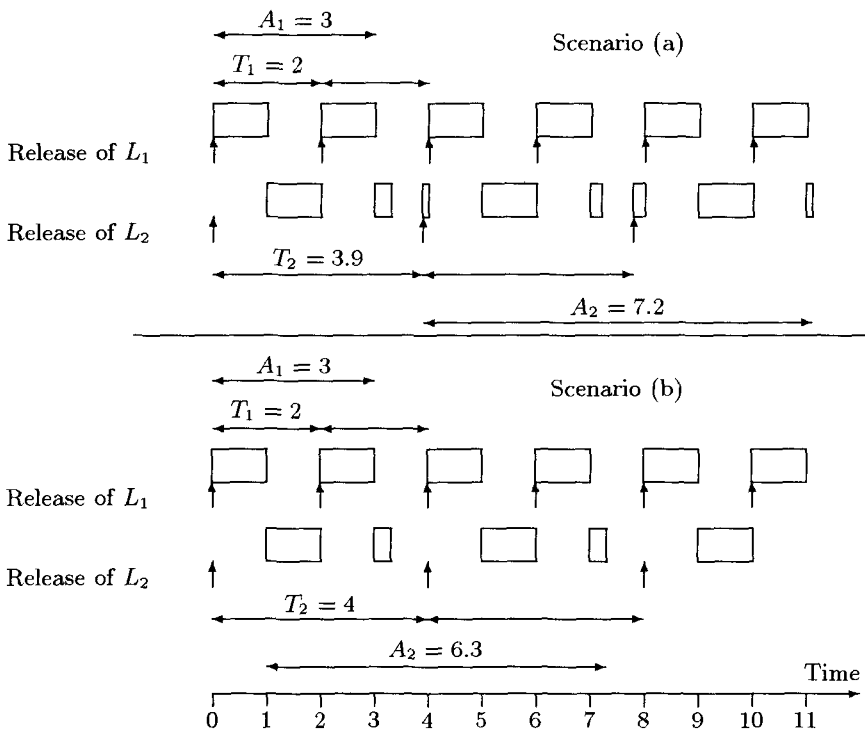

Fig. 3. In scenario (a) there are two processes L_1 and L_2. Process L_1 has a period $T_1 = 2$ and a computation time $C_1 = 1$. Process L_2 has a period $T_2 = 3.9$ and a computation time $C_2 = 1.3$. In this scenario we are able to meet the age constraints $A_1 = 3$ and $A_2 = 7.2$, i.e. $A_2 = T_2 + R_2 = T_2 + C_2 + 2C_1 = 3.9 + 1.3 + 2 = 7.2$. In scenario (b) we consider the same processes, with the exception that the period of L_2 has been extended, i.e. $T_2 = 2T_1 = 4$. The phasing of L_1 and L_2 has also been adjusted such that a release of L_2 always coincides with a release of L_1. In this scenario we are able to meet the age constraints $A_1 = 3$ and $A_2 = 6.3$, i.e. $A_2 = T_2 + R_2 = T_2 + C_2 + C_1 = 4 + 1.3 + 1 = 6.3$. Consequently, by extending the period of L_2 we were able to meet tougher age constraints.

Theorem 4. *A set of processes which is schedulable using the unoptimized periods T_i and an arbitrary phasing of processes is also schedulable using the optimized periods t_i, provided that we are able to adjust the phasing of the processes.*

Proof. Let r_i denote the maximum response time for process L_i using the periods t_i and the optimized phasing. Obviously, $T_i \leq t_i$. From this we conclude that $r_i = C_i + \sum_{j=1}^{i-1} \lceil r_i/t_j \rceil C_j \leq R_i = C_i + \sum_{j=1}^{i-1} \lceil R_i/T_j \rceil C_j$. Consequently, if $R_i \leq A_i/2$, then $r_i \leq A_i/2$.

Figure 4 shows a process set which is schedulable using the improved version of our scheme. This process set would not have been schedulable using the unoptimized version of our scheme. Consequently, the improved version of the scheme is better in the sense that we are able to schedule a larger number of process sets.

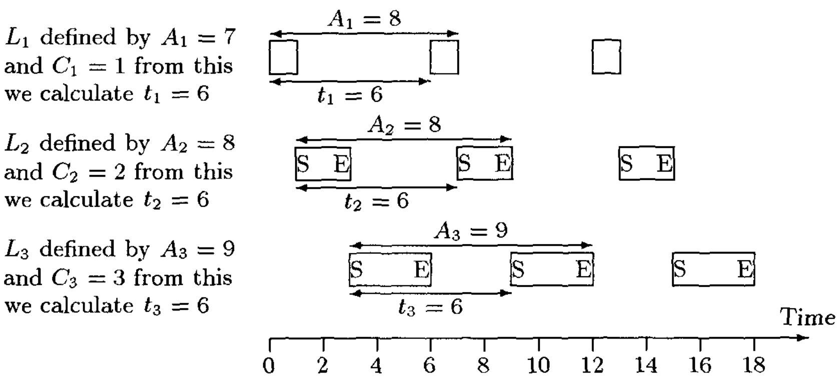

Fig. 4. Three processes which are schedulable using the improved scheme.

5 Conclusions

In this paper a method for scheduling age constraint processes has been presented. An age constraint process is defined by two values: the maximum time between the beginning of the process' execution in cycle k to the end of the process' execution in cycle $k+1$ (the age constraint), and the maximum computation time in each cycle, i.e. the period of each process is not explicitly defined. We present an algorithm for calculating process periods. Once the periods have been calculated the process set can be executed using preemption and fixed priority scheduling. The priorities are defined by the rate-monotone algorithm.

Trivially, the age constraint can be met by using periods equal to half the age constraint. However, the periods obtained from our method are better in the sense that they make it possible to schedule a larger number of process sets than we would have been able to do if we had used periods equal to half the age constraint. All process sets which are schedulable using periods equal to half the age constraint are also schedulable using our method.

A simple analysis of our method shows that the rate-monotone priority assignment algorithm is optimal for all process sets containing two processes, using our process periods. We also show that all process sets with two processes, for which $C_1/(A_1-C_1)+C_2/(A_2-C_2)$ is less than $2(\sqrt{2}-1) = 0.83$, are schedulable using our scheme. There are, however, process sets containing two processes for which $C_1/(A_1 - C_1) + C_2/(A_2 - C_2) = 2(\sqrt{2} - 1) + e$ (for any $e > 0$) which are not schedulable using our scheme, i.e. we provide a simple schedulability test for process sets containing two processes.

We also define an improved version of our method. The improved version capitalizes on the fact that there is room for optimization when the period of one process is an integer multiple of the period of another process. The improved version of the method is better than the original version in the sense that we are able to schedule a larger number of process sets. All process sets which are schedulable using the original version are also schedulable using the improved version.

Previous work on age constraint process have concentrated on creating cyclic interleavings of processes [1]. Other studies have looked at age constraint pro-·esses which communicate [5]. One such scenario is scheduling of age constraint ocesses in the context of hard real-time database systems [2].

References

1. W. Albrecht, and R. Wisser, "Schedulers for Age Constraint Tasks and Their Performance Evaluation", In Proceedings of the Third International Euro-Par Conference, Passau, Germany, August 1997.
2. N. C. Audsley, A. Burns, M.F. Richardson, and A.J. Wellings, "Absolute and Relative Temporal Constraints in Hard Real-Time Databases", In Proceedings of IEEE Euromicro Workshop on Real-Time Systems, Los Alamitos, California, February 1992.
3. A. Burns, and A. Wellings, "Real-Time Systems and Programming Languages", Addison-Wesley, 1996.
4. J.P. Lehoczky, L. Sha, and Y. Ding, "The rate monotone scheduling algorithm: exact characterization and average case behavior", In Proceedings of IEEE 10th Real-Time Systems Symposium, December 1989.
5. X. Song, and J. Liu, "How Well can Data Temporal Consistency be Maintained", In Proceedings of the 1992 IEEE Symposium on Computer Aided Control Systems design, Napa, California, USA, March 1992.