

An Exact Execution Time Test for Fixed Priority Real -Time Tasks

Hadi M. Tjandrasa
Systems Software Engineer
Lockheed Martin Space Operations
At the Johnson Space Center
hadi.m.tjandrasa1@jsc.nasa.gov
htjandra@computer.org

Sadegh Davari and Ted Leibfried
Division of Computing and Mathematics
University of Houston-Clear Lake
Houston, TX 77058
(281) 283-3860
{Davari, Leibfried}@cl.uh.edu

Abstract

In most practical systems, periodic tasks commonly contribute to the major parts of application tasks in the systems. Under preemptive scheduling, priorities are assigned to the periodic tasks. Task priorities can be static, which remains constant, or dynamic, which will change, during the life of the system. A preemptive static priority scheduling algorithm possesses predictable behavior for which, given a set of periodic tasks, the schedulability can be well analyzed. In this paper, an exact schedulability analysis, called an exact execution time test, for a set of periodic tasks scheduled by a static priority scheduling algorithm will be introduced and investigated.

1. Introduction

Task scheduling in real-time systems can be classified into two categories according to task characteristics: preemptive [5,8] and non-preemptive [4] scheduling algorithms. The preemptive scheduling algorithm based on priorities can be static or dynamic. In a static priority scheduling algorithm, task priorities are predefined and remain fixed during execution, from request to request. For tasks scheduled by a dynamic priority scheduling algorithm, their priorities may change during execution.

We distinguish three approaches to real-time scheduling algorithms: heuristic, deterministic and predictable.

In heuristic scheduling approach the goal is to maximize performance. This approach

generally lacks flexibility and is tailored to a specific application and environment.

The deterministic scheduling algorithm requires task characteristics to be predefined a priori. An example of deterministic scheduling algorithm is the cyclic executive [1,2]. A cyclic executive is a scheduling algorithm that interleaves the executions of periodic tasks in deterministic fashion. The cyclic schedule is comprised of a major cycle and several minor cycles. The major cycle is defined to be the least common multiple of the periods of all periodic tasks in the task set. For a major cycle to exist, the frequencies of all tasks must be harmonically related. That is, the period of the minor cycle is equal to the greatest common divisor of the periods of all tasks in the task set. Additional discussions on scheduling of nonharmonic tasks under the cyclic executive can be found in [6].

In real-time systems, the system is predictable if all tasks can meet their deadlines at all periodic requests. Hence, the initiation and completion time of all tasks relative to their periodic arrival time are not necessary to be known a priori. A static scheduling algorithm that can predict the schedulability of a given set of periodic tasks is the rate monotonic algorithm [5]. The rate monotonic scheduling policy defines task priorities according to their periodic request rates. Tasks with higher request rates are assigned higher priorities. The scheduling algorithm is optimal in the sense that a set of tasks can be scheduled by the rate monotonic algorithm if it can be scheduled by any static priority scheduling algorithm. Further detailed discussion and analysis of this scheduling algorithm can be found in [3,7,9].

2.0 Schedulability Analysis

Given a set of periodic tasks scheduled by a static priority scheduling algorithm, Liu and Layland [5] defined and proved that a critical instant for any task occurs when all tasks in the set are requested simultaneously. That is, the worst-case condition for processor utilization occurs at the early region of time span when all task requests arrive simultaneously. On the notion of critical instant, therefore, the schedulability of periodic tasks scheduled by a static priority scheduling algorithm need only be analyzed for the time span of the period of a task whose frequency is the lowest in the set, i.e., largest period.

2.1 Task Characteristics and Notations

The schedulability analysis presented in this paper assumes the following characteristics of the periodic tasks:

1. Task priorities are static. Priorities do not change during real-time execution.
2. Tasks are preemptible but do not suspend themselves during execution.
3. Deadlines are at the end of periods.
4. No precedent constraints exist between tasks (this issue is not within the scope of this paper).
5. Task periods are constant between requests.
6. Context switches and system overhead are not accounted for.

The notations are given as follows:

n : number of periodic tasks to be scheduled.

τ_i : task i , for $i = 1, 2, \dots, n$.

T_i : request period of τ_i .

P_i : priority of τ_i .

C_i : worst-case computation time of τ_i in a period T_i .

E_i : cumulative worst-case execution (computation) time of τ_i within the time interval of one period of the lowest priority task in the task set.

E : total execution (computation) time for all tasks within the time interval of one period of the task whose priority is the lowest in the task set.

Task with higher priorities are given smaller task numbers, i.e., $P_1 > P_2 > \dots > P_n$.

2.2 Exact Execution Time for Two Periodic Tasks

Let us begin by investigating a schedule of two periodic tasks. The total worst-case execution time of τ_1 and τ_2 , with simultaneous request times and $P_1 > P_2$, for one period T_2 is defined as

$$E = E_1 + E_2.$$

Task 1:

The execution time of τ_1 for one period T_2 depends upon the number of arrivals of task τ_1 within the period T_2 and the minimum function can be shown to be

$$E_1 = \min(\lceil T_2/T_1 \rceil C_1, (\lfloor T_2/T_1 \rfloor C_1 + T_2 - \lfloor T_2/T_1 \rfloor T_1)).$$

Let $X = \lceil T_2/T_1 \rceil C_1$,

and $Y = \lfloor T_2/T_1 \rfloor C_1 + T_2 - \lfloor T_2/T_1 \rfloor T_1$.

One of the following three cases may occur with respect to the values of X and Y :

Case 1: $X = Y$.

The equality exists either:

- a) when T_2 is an integral multiple of T_1 (i.e., $T_2 - \lfloor T_2/T_1 \rfloor T_1 = 0$), or
- b) when the $\lceil T_2/T_1 \rceil^{\text{th}}$ execution of τ_1 completes exactly at the end of T_2 .

Case 2: $X > Y$.

It occurs when the second periodic request of τ_2 overlaps the execution of the $\lceil T_2/T_1 \rceil^{\text{th}}$ execution of τ_1 .

Case 3: $X < Y$.

This condition exists when the relationships between T_1 and T_2 described in Case 1 and Case2 are not present.

Note that the value of C_2 is irrelevant to the schedulability of τ_1 since $P_1 > P_2$. That is, τ_1 will get executed whenever its priority becomes active.

Task2:

Assuming that the task set is schedulable, the execution time E_2 of τ_2 for one period T_2 is simply equal to C_2 . A similar minimum function used in computing E_1 can be applied in computing E_2 , as follows:

$$E_2 = \min(\lceil T_2/T_2 \rceil C_2, (\lfloor T_2/T_2 \rfloor C_2 + T_2 - \lfloor T_2/T_2 \rfloor T_2))$$

In this case, the first argument of the minimum function is equivalent to the second argument, and both are equal to C_2 .

2.3 Exact Execution Time for a Set of Periodic Tasks

We now investigate further in deriving the equation for calculating the total exact execution time, E , for a set of tasks.

The total execution time for a schedulable set of n tasks is defined as

$$E = E_1 + E_2 + \dots + E_{n-1} + E_n \leq T_n.$$

where, for $i = 1, 2, \dots, n$,

$$E_i = \min(\lceil T_n/T_i \rceil C_i, (\lfloor T_n/T_i \rfloor C_i + T_n - \lfloor T_n/T_i \rfloor T_i - \sum_{j=1}^{i-1} (A_j - B_j)))$$

and

$$A_j = \min(\lceil T_n/T_j \rceil C_j, (\lfloor T_n/T_j \rfloor C_j + T_n - \lfloor T_n/T_j \rfloor T_j - \sum_{a=1}^{j-1} (A'_a - B'_a))),$$

$$\alpha = \lfloor T_n/T_i \rfloor T_i,$$

$$B_j = \min(\lceil \alpha/T_j \rceil C_j, (\lfloor \alpha/T_j \rfloor C_j + \alpha - \lfloor \alpha/T_j \rfloor T_j - \sum_{a=1}^{j-1} (A'_a - B'_a))).$$

Note that the above equation, minimum function, is an extension of the equation discussed in the previous section, with the addition of the term $\sum (A_j - B_j)$ to the equation. $\sum (A_j - B_j)$ is the total execution time of all τ_j , $1 \leq j < i$, within the time interval between the final request time of τ_i (in T_n) and the end of T_n . It can be noted that E_i , A_j and B_j have similar forms and terms of expressions, and the expressions of A'_a and B'_a correspond to A_j and B_j , respectively. Hence, they can be represented by a recursive function as given in [11].

For schedulability analysis of a set of n tasks, an execution time test must be conducted for each task. The execution time test for each set of k tasks, for $1 \leq k \leq n$, is necessary because the total execution time of a set of n tasks may be less than or equal to T_n , but the total execution time of a set of $(n-i)$ tasks, for $i = 1, 2, \dots, n-1$, may be greater than T_{n-i} .

From our previous investigations of the two-task, we can generalize and apply the analytical results into a set of n tasks and by induction this process results in the following theorem, which is called *exact execution time test*.

Theorem 1:

A set of n independent periodic tasks, with deadlines at end of periods and $P_1 > P_2 > \dots > P_n$, is schedulable, for all task phasings under a static priority scheduling algorithm, if and only if,

$$\forall k, 1 \leq k \leq n, \quad E = \sum_{i=1}^k E_i \leq T_k,$$

where

$$E_i = \min(\lceil T_k/T_i \rceil C_i, (\lfloor T_k/T_i \rfloor C_i + T_k - \lfloor T_k/T_i \rfloor T_i - \sum_{j=1}^{i-1} (A_j - B_j))).$$

With

$$a) \quad A_j = \min(\lceil T_k/T_j \rceil C_j, (\lfloor T_k/T_j \rfloor C_j + T_k - \lfloor T_k/T_j \rfloor T_j - \sum_{a=1}^{j-1} (A'_a - B'_a))),$$

where

$$A'_a = \min(\lceil T_k/T_a \rceil C_a, (\lfloor T_k/T_a \rfloor C_a + T_k - \lfloor T_k/T_a \rfloor T_a - \sum_{b=1}^{a-1} (A''_b - B''_b))),$$

and defining

$$\alpha' = \lfloor T_k/T_j \rfloor T_j,$$

$$B'_a = \min(\lceil \alpha'/T_a \rceil C_a, (\lfloor \alpha'/T_a \rfloor C_a + \alpha' - \lfloor \alpha'/T_a \rfloor T_a - \sum_{b=1}^{a-1} (A''_b - B''_b))).$$

$$b) \quad \text{Also define } \alpha = \lfloor T_k/T_i \rfloor T_i,$$

and

$$B_j = \min(\lceil \alpha/T_j \rceil C_j, (\lfloor \alpha/T_j \rfloor C_j + \alpha - \lfloor \alpha/T_j \rfloor T_j - \sum_{a=1}^{j-1} (A'_a - B'_a))),$$

where

$$A'_a = \min(\lceil T_k/T_a \rceil C_a, (\lfloor T_k/T_a \rfloor C_a + T_k - \lfloor T_k/T_a \rfloor T_a - \sum_{b=1}^{a-1} (A''_b - B''_b))),$$

and define

$$\alpha' = \lfloor \alpha/T_j \rfloor T_j,$$

$$B'_a = \min(\lceil \alpha'/T_a \rceil C_a, (\lfloor \alpha'/T_a \rfloor C_a + \alpha' - \lfloor \alpha'/T_a \rfloor T_a - \sum_{b=1}^{a-1} (A''_b - B''_b))).$$

And the expressions of the A'_a and B'_a functions correspond to the recursive definitions of the A_j and B_j functions, respectively.

Assuming the inequality condition ($\forall k, 1 \leq k \leq n, E \leq T_k$) is satisfied, then the exact processor utilization factor, U , of a schedulable set of n tasks is simply defined as $U = E/T_n \leq 1$.

The theorem proof is demonstrable by carrying the inductive process further. With the recursion and iteration that are inherent in the equation, the usage of the execution time test will become much easier by implementing it as a program [11].

In this context, a set of periodic tasks is *schedulable* if all tasks in the set can meet their deadlines at the critical instant.

3. Conclusion

We have investigated the schedulability of periodic tasks scheduled by a static priority scheduling algorithm and introduced a schedulability test called the exact execution time test. The execution time of every task is computed for the period of the lowest priority task in the set in order to determine the schedulability of the task set. The exact execution time test provides exact and optimistic analytical results.

For a given set of n tasks ($\forall k, 1 \leq k \leq n, i = 1, 2, \dots, n$) at their critical instants, the test provides the following useful analytical results:

- exact execution time E_i in one period T_k .
- processor idle time in one period T_k , if any, which is defined as $(T_k - E)$.
- execution time overrun in one period T_k , if any, which is defined as $(E - T_k)$.

Timing requirements of periodic tasks in many real-time systems can be satisfied by ensuring the predictability of meeting the deadlines rather than by employing determinism. The exact execution time test can be utilized as a valuable tool in predicting the schedulability of periodic tasks scheduled by static priority scheduling algorithm.

References

- [1] Baker, T.P. and Shaw, A., "The cyclic Executive Model and Ada", *Proc. IEEE Real-time Systems Symposium*, 1988, pp. 120-129.
- [2] Carlow, G.D., "Architecture of the Space Shuttle Primary Avionics Software System", *Communication of the ACM*, September 1984, pp. 926-936.
- [3] Davari, S., "Rate Monotonic Scheduling Theory" - tutorial, *RICIS Symposium*, University of Houston-Clear Lake, Houston, Texas, 1992.
- [4] Jeffay, K., Stanat, D.F., and Martel, C.U., "On Non-preemptive Scheduling of Periodic and Sporadic Tasks", *Real-time Systems Symposium*, 1991, pp.129-139.
- [5] Liu, C.L. and Layland, J.W., "Scheduling Algorithm for Multiprogramming in Hard Real-time Environment", *Journal of the ACM*, 20, January 1973, pp. 46-61.
- [6] Locke, D.C., "Software Architecture for Hard Real-Time Applications: Cyclic Executive vs. Fixed Priority Execution", *The Journal of Real-time Systems*, 4, 1992, pp. 37-53.
- [7] Sha, L., Lehoczky, J., and Ding, Y., "The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior", *IEEE Real-time Systems Symposium*, 1989, pp.166-171.
- [8] Sha, L., Lehoczky, J., and Rajkumar, R., "Solution for Some Practical Problems in Prioritized Preemption Scheduling", *IEEE Real-time Systems Symposium*, 1986, pp. 181-191.
- [9] Sha, L. and Sathaye, S.S., "A Systematic Approach to Designing Distributed Real-Time Systems", *IEEE Computer*, 26, 1993, pp. 181-191.
- [10] Stankovic, A. and Ramamritham, K., "What is Predictability for Real-time Systems?", *J. Real-time Systems*, Vol. 2, December 1990, pp.247-254.
- [11] Tjandra, H.M., "A Practical Approach to a Static Priority Scheduling: Schedulability Analysis and Jitter on Uniform Sampling Tasks", *Master Thesis*, University of Houston-Clear Lake, Houston, Texas, 1994.