

# Handling Jitter in Uniform Sampling Tasks

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

Sadegh Davari and Ted Leibfried  
Division of Computing and Mathematics  
University of Houston-Clear Lake  
Houston, TX 77058  
(281) 283-3865  
{Davari, Leibfried}@cl.uh.edu

## Abstract

*In real-time systems, stringent timing requirements may impose on periodic tasks to be initiated immediately upon the arrivals of their periodic requests or at specific time relative to their periodic requests. One of the reasons for the requirements is to avoid jitter. In this paper, we propose treatments to the jitter problems that may occur on scheduling uniform sampling tasks and introduce a schedulability analysis for a task set consisting of periodic tasks and a group task, which is a collection of data acquisition subtasks, scheduled by a static priority scheduling algorithm.*

## 1.0 Introduction

In some real-time systems, stringent timing requirements may exist calling for periodic tasks to be initiated immediately upon the arrivals of their periodic requests. One of the reasons for these requirements is to avoid jitter. Jitter can be attributed to variations in time of occurrence of successive instances of events. Examples of these events are input-output operation, sampling of a continuous signal and computation time of a periodic task.

In this paper, we investigate jitter associated with the schedule of uniform sampling tasks and propose solutions to alleviate the jitter problem. It is assumed that the execution time of data acquisition segments of uniform sampling tasks are short. We define a uniform sampling task as a periodic task whose function consists of a data acquisition segment and a processing segment which processes the sampled data according to a specific algorithm. Uniform sampling processes are widely utilized in real-life applications such

as in process controls, environmental monitoring and in real-time simulation such as space, flight or ground vehicle simulation.

Under the rate monotonic scheduling [1,3,4], which is a static priority scheduling algorithm, tasks in a set are scheduled according to their active priorities; therefore, uniform sampling tasks are susceptible to severe jitter unless they are assigned very high priorities. It is the primary intention of this paper to investigate and to utilize priority scheme in treating jitter problem associated with uniform sampling tasks. We will also introduce a schedulability analysis, called *modified execution time test*, for a task set consisting of periodic tasks and a group task, which is a collection of data acquisition subtasks, scheduled by a static priority scheduling algorithm.

There are three alternatives to the scheduling of uniform sampling tasks to be investigated. They are:

1. A group task consisting of data acquisition subtasks.
2. Dual priority uniform sampling tasks.
3. A schedule of uniform sampling tasks with fixed phasings.

In this paper we are only investigating alternative 1.

One proposed technique to alleviate the jitter problem is by grouping the data acquisition segments of all uniform sampling tasks into one separate task, called a group task, which is assigned a very high priority. Splitting the data acquisition subtasks from the processing subtasks is discussed in the details of the rate-group model [2].

In grouping data acquisition subtasks, one of the following three possibilities may occur with respect to the periods of uniform sampling tasks:

- All uniform sampling tasks have equal periods or frequency rates.
- Periods of uniform sampling tasks are harmonically divisible.
- Uniform sampling tasks have non-harmonic periods.

The periods of tasks in a set are said to be harmonically divisible if each task frequency is an integral multiple to all lower frequencies in the set. In this paper we will only investigate the case of uniform sampling tasks with non-harmonic periods.

The following notations will be used in the analysis:

- $\tau_i$  : Task  $i$ , for  $i = 1, 2, \dots, n$ .
- $T_i$  : Request period of  $\tau_i$ .
- $C_i$  : Total computation time of  $\tau_i$  ( $C_{p,i} + C_{d,i}$ ).
- $E_i$  : execution (computation) time of  $\tau_i$  within the time interval of one period of the task whose priority is the lowest in the task set.
- $C_{d,i}$ : Worst-case computation time of the data acquisition subtask (segment) of  $\tau_i$ .
- $C_{p,i}$ : Worst-case computation time of the processing subtask of  $\tau_i$ .
- $P_{d,i}$ : Priority of the data acquisition subtask of  $\tau_i$ .
- $P_{p,i}$ : Priority of the processing subtask of  $\tau_i$ .  
Note that  $P_i$  will be interchangeably used.
- $\tau_g$  : Periodic group task consisting of data acquisition subtasks.
- $T_g$  : Request period of  $\tau_g$ .
- $P_g$  : Priority of  $\tau_g$ .

## 2.0 Uniform Sampling Tasks with Non-harmonically Divisible Frequencies

Non-harmonically divisible frequencies refer to the task frequencies in a task set whose frequencies are neither equal nor harmonically divisible. We will investigate the schedule of data acquisition subtasks contained in a group task and define the limitation with regard to its computation time, and then briefly discuss the least upper bound test.

### 2.1 Maximum Bound on Computation Time of Data Acquisition Subtasks

Frame overlaps, occurring in a schedule of data acquisition subtasks whose frequencies are not harmonically divisible, impose a stringent limitation to the aggregate computation time of

all data acquisition subtasks in the group task. That is, in order to generate constant intervals consistently between successive subtask executions, it is necessary that the aggregate computation time of the data acquisition subtasks in a group task be bounded by the greatest common divisor (GCD) of the periods of contributing data acquisition subtasks; the periods are originated from and equivalent to the periods of their respective uniform sampling tasks. Let us rephrase the statement precisely in the following property.

#### Property 1:

For a set of uniform sampling tasks, whose periodic requests are not harmonically divisible, that are initiated simultaneously at the start (time 0), the aggregate computation time of all subtasks in the group task must be less than or equal to the GCD of the periods of their respective uniform sampling tasks in order to ensure constant intervals between successive executions of the subtasks. This necessary condition is defined as follows:

$$\sum_{i=1}^n C_{d,i} \leq \text{the GCD of } \{T_i \mid \tau_i \text{ is a uniform sampling task}\}.$$

#### *Proof:*

This bound is correlated to the fact that the smallest difference (greater than 0) of time span between two adjacent scheduling points of uniform sampling tasks is equivalent to the GCD of the periods of uniform sampling tasks, assuming their first periodic requests arrive simultaneously at time 0. That is, the arrivals of all tasks' periodic requests will not fall within any time span delimited by two scheduling points.

## 3.0 Schedulability Analysis of a Task Set with a Group Task

The *exact execution time test* is a schedulability analysis for a set of periodic tasks scheduled by a static priority scheduling algorithm. It is presented in a companion WIP paper, "An Exact Execution Time Test for Fixed Priority Real-Time Tasks".

With the addition of a group task to the task set, the exact execution time test needs to be extended to consider the schedule of the data acquisition subtasks contained in the group task.

### 3.1 Modified Execution Time (MET) Test

The *exact execution time test* given in [7] is extended here in order to accommodate the addition of the group task to the task set. This is necessary because of the segmentation of the aggregate computation time of the subtasks within the group task. The extended version of the execution time test, called the *modified execution time (MET) test*, will be introduced in the following theorem.

**Theorem 1:**

For  $n$  periodic tasks, in a task set with uniform sampling tasks and  $P_1 > P_2 > \dots > P_n$ , that are initiated simultaneously at the start, time 0, and  $T_g = \text{Min}(\{T_i \mid 1 \leq i \leq n, \tau_i \text{ is a uniform sampling task}\})$ , then the task set can be scheduled by a static priority scheduling algorithm, if the following conditions are satisfied:

- a) *Property 1* holds.
- b)  $\forall k, 1 \leq k \leq n$ ,

$$E = \sum_{u=1}^n C_{d,u} + \sum_{v=1}^{k-1} (\lceil T_k/T_v \rceil - 1) C_{d,v} + \sum_{i=1}^k E_i \leq T_k$$

( $E_i$  is defined in [7])

**Proof:**

For a set of periodic tasks scheduled by a static priority scheduling algorithm, it was proved in [3] that the critical instant for any task occurs whenever the task is requested simultaneously with requests for all higher priority tasks. Therefore, the busiest time span of the processor utilization will be in the region following and closer to the point of the critical instant. This is the premise that underlies the following arguments.

With *property 1* and  $T_g$  defined above,  $\sum_{u=1}^n C_{d,u}$

is the aggregate computation time of all data acquisition subtasks for the first periodic request of  $\tau_g$ . This aggregate computation time must first be obtained because  $P_g$  is the highest priority in the task set. For the executions of all data acquisition subtasks after the second arrival of  $\tau_g$ , the total execution time within the interval between the second arrival of  $\tau_g$  and the end of  $T_k$  is defined by

$$\sum_{v=1}^{k-1} (\lceil T_k/T_v \rceil - 1) C_{d,v}.$$

The total execution time of  $k$  tasks, excluding  $\tau_g$ , within the time interval between time 0 and the end of

$T_k$ , is defined by  $\sum_{i=1}^k E_i$  in [7].

The exact execution time test for each task,  $\tau_k, 1 \leq k \leq n$ , and  $P_1 > P_2 > \dots > P_n$ , implies that if the first  $C_k$  can be satisfied before the second arrival of  $\tau_k$ , i.e.,  $C_k$  can be completed prior to the first deadline at its critical instant, then  $\tau_k$  can be scheduled by a static priority scheduling algorithm. The execution time test also implies that if  $\tau_k$  is schedulable then  $\tau_i$ , for  $1 \leq i < k$ , is also schedulable.

This theorem is only necessary when the task periods are non-harmonically divisible.

### 3.2 Example of the MET Test

As an example, consider a set of three tasks with two uniform sampling tasks illustrated in Figure 1. Table 1 lists the tasks' parameters in which  $P_{d,1} > P_{d,2} > P_{p,1} > P_{p,2} > P_{p,3}$ .

$\tau$	$T$	$P_d$	$P_p$	$C_d$	$C_p$
1	30	1	3	2	5
2	45	2	4	3	8
3	90	-	5	-	10

Table 1: Three tasks with two uniform sampling tasks.

**MET test:**

- a) The GCD is 15.

$$2 + 3 \leq 15 ? \text{ Answer: Yes.}$$

- b)  $k=1$ :

$$E = 5 + 0 + E_1 \leq T_1 ?$$

$$E = 5 + 0 + 5 \leq 30 ? \text{ Answer: Yes.}$$

- $k=2$ :

$$E = 5 + (\lceil 45/30 \rceil - 1)2 + E_1 + E_2 \leq T_2 ?$$

$$E = 5 + 2 + 10 + 8 \leq 45 ? \text{ Answer: Yes.}$$

- $k=3$ :

$$E = 5 + (\lceil 90/30 \rceil - 1)2 + (\lceil 90/45 \rceil - 1)3 + E_1 + E_2 + E_3 \leq T_3 ?$$

$$= 5 + 4 + 3 + 15 + 16 + 10 \leq 90 ?$$

Answer: Yes.

Since all inequalities are satisfied, the task set can be scheduled by a static priority scheduling algorithm.

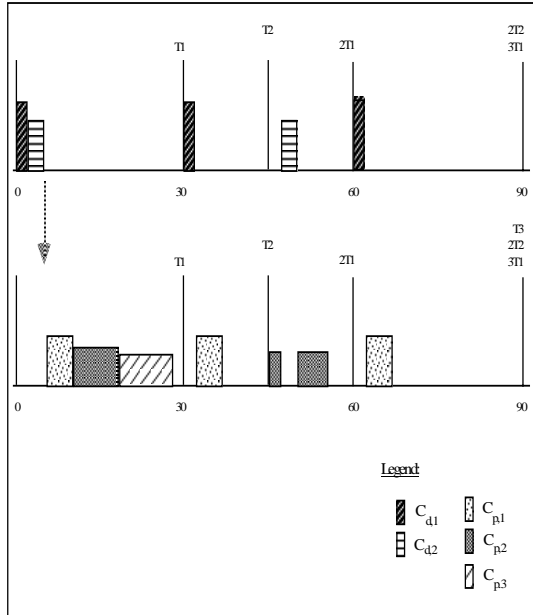


Figure 1: Schedule of a task set with two uniform sampling tasks.

## 4.0 Conclusion

We are investigating the techniques of scheduling a task set containing uniform sampling tasks. One of the techniques, the group task model, requires that the GCD bound must be satisfied to ensure constant intervals between successive executions of the data acquisition subtasks. The modified execution time (MET) test introduced in this paper can predict the schedulability of a task set and provide useful results for analysis. It can predict the schedulability of a task set better than the least upper bound test, since the result of the least upper bound test is very conservative or pessimistic.

## References

- [1] Davari, S., "Rate Monotonic Scheduling Theory" - tutorial, *RICIS Symposium*, University of Houston-Clear Lake, Houston, Texas, 1992.
- [2] Kim, M.J., "Hard Real-time Scheduling for the Rate-group Model", Ph.D. Dissertation, Texas A&M University, College Station, Texas, 1992.

- [3] 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.
- [4] 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.
- [5] Sha, L., Rajkumar, R., and Lehoczky, J.P., "Priority Inheritance Protocols - An Approach to Real-time Synchronization", Carnegie Mellon University, 23 May 1988.
- [6] 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.
- [7] Tjandra, H.M., Davari, S., and Leibfried T.F., "An Exact Execution Time Test for Fixed Priority Real-Time Tasks". 21<sup>st</sup> IEEE RTSS, WIP Proceedings, Nov. 2000, Orlando, Florida.