

# Worst Case Response Time Analysis of Sporadic Tasks with Precedence Constrained Subtasks Using Non-preemptive EDF Scheduling

Armaghan Darbandi and Myung Kyun Kim

**Abstract** In this paper, we address the problem of Worst Case Response Time (WCRT) analysis of hard real-time systems composed of sporadic tasks scheduled under non-preemptive Earliest Deadline First (EDF) scheduler. WCRT of each subtask is examined based on the constructed Directed Acyclic Graph of each task. Previous work, does not consider arbitrary timing requirements of subtasks; authors investigate simultaneous arrival time of subtasks only where arbitrary deadlines is not concerned. In contrast to previous work that provides guarantee for a task set that in fact is not schedulable in the worst case, our results show accurate WCRTs.

**Keywords** Worst Case Response Time · EDF · Non-preemptive

## 1 Introduction

In this paper, we consider the problem of schedulability of a set of independent sporadic tasks, where each sporadic task consists of a set of precedence constrained subtasks with arbitrary timing requirements. Subtasks are scheduled using non-preemptive EDF scheduler such that; subtasks with precedence constraints are scheduled according to their precedence relations, subtasks with no precedence relations are scheduled according to non-preemptive EDF algorithm.

Spuri et al. [1] models Worst Case Response Time (WCRT) for sporadically periodic tasks with arbitrary deadlines. However, [1] does not cover subtask

---

A. Darbandi · M. K. Kim (✉)

School of Electrical Engineering, University of Ulsan, Nam-Gu Ulsan, South Korea  
e-mail: mkkim@ulsan.ac.kr

dependencies. One approach to consider precedence constraints between subtasks of a sporadic task is to represent the sporadic task by a graph [2, 3]. In order to find WCRT of a subtask, those approaches adopt a technique to convert the sporadic graph of the task under analysis to a canonical chain. This technique is still somehow an approximation to the response times, because in some particular cases, transformation of graph to chain is not able to exploit sufficiently the priorities among subtasks. Moreover, authors covered simultaneous arrival time of subtasks only. In this paper, to acquire accurate WCRTs, no deadline modification is allowed and the analysis is performed on the sporadic graph. In the study of task model defined in [3] (simultaneous arrival of subtasks), our contribution is shown to be effective with an extensive set of simulations in Sect. 5.

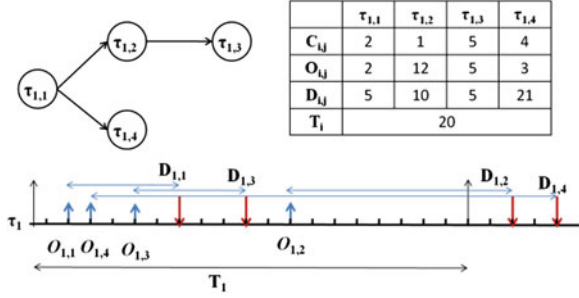
This paper starts by introducing related works in Sect. 2, the computational model in Sect. 3. The proposed WCRT analysis is described in Sect. 4. A case study is presented in Sect. 5. Finally, our conclusions are stated in Sect. 6.

## 2 Related Works

A well-known result for periodic tasks is that the preemptive EDF algorithm is optimal in the sense that it will successfully generate a processor schedule for any periodic task system that can be scheduled [4]. In contrast to periodic tasks, a sporadic task is invoked at arbitrary times but with a specified minimum time interval between invocations. Spuri in [1] proposed WCRT analysis of sporadically periodic task sets with arbitrary timing requirements scheduled under preemptive EDF. However, [1] does not cover dependency among subtasks of a sporadic task. The approaches proposed by Zhao in [2, 3], concerns WCRT analysis for a set of sporadic tasks, scheduled under non-preemptive EDF and fixed priority, respectively. The main idea in [3] is to transform the sporadic graph of analyzed task to a canonical chain so that each subtask in the obtained canonical chain has a priority strictly higher than its predecessors. However, authors in [2, 3] cover simultaneous arrival of subtasks only. Since our approach allows arbitrary arrival time of subtasks within a task, our WCRT analysis includes the computational model presented by [3] as a special case. In the study of task level precedence constraints, [5] generalizes the concept of precedence constraint between tasks to allow tasks to have different rates. [6] partitions the graph of tasks into a set of parallel flows, to meet the specified timing constraints.

As for distributed hard real-time systems, Spuri in [7] adapted the Holistic Analysis technique to the systems that scheduled under EDF. Later [8–11] improved the estimations of WCRTs by developing the offset based technique. The fundamental principle behind is that, given the offset information of tasks at a communication resource one can compute (in a worst-case manner) the jitter and offset for tasks leaving that communication resource, which in turn becomes the arrival pattern for tasks to the subsequent communication resource [12]. Another extension is proposed by [13], which allows each task to have one or more immediate successors.

**Fig. 1** Computational model of a sporadic task



### 3 System Model and Notations

**Definition Sporadic Task** ( $\tau_i$ ) sporadic task  $\tau_i$  consists of  $n_i$  number of subtasks  $\tau_i = \{\tau_{i,1}, \dots, \tau_{i,n_i}\}$  with precedence constraints and arbitrary timing requirements. Inter-arrival time of each task is separated by a minimum time  $T_i$ . Each task activation at time  $t$  results in activation of all subtasks, each one at time  $t_{i,j} = t + O_{i,j}$ , where  $O_{i,j}$  denotes the arrival time relative to the activation of the task, Fig. 1. Each subtask has an execution time  $C_{i,j}$ , and relative deadline  $D_{i,j}$ . The absolute deadline  $d_{i,j}$  of subtask  $\tau_{i,j}$  requested at time  $t_{i,j}$  equals  $d_{i,j} = t_{i,j} + D_{i,j}$ . Relative deadline of task  $\tau_i$  is denoted by  $D_i$ ; where  $D_i$  equals  $D_i = \max_j(O_{i,j} + D_{i,j})$ . The precedence relations in each sporadic task are represented by a Directed Acyclic Graph (DAG) as shown in Fig. 1.

**Definition Busy Period** the time interval delimited by two successive idle times, during which the CPU is not idle. Let us denote the length of the longest busy period by  $L$  and the starting instance of the longest busy period by  $t_B$ .

**Definition Deadline-  $d$  Busy Period** A processor busy period in which only the subtasks with absolute deadline smaller than or equal to  $d$  execute.

**Definition Worst Case Response Time (WCRT)** ( $R_{i,j}$ ) maximum possible delay between arrival time and completion time of subtask  $\tau_{i,j}$  on each activation time of the subtask  $\tau_{i,j}$  inside the longest busy period:

$$R_{i,j} = \max\{R_{i,j}(r)\}; \forall r t_B \leq r < t_B + L \tag{1}$$

Procedure 1 determines the order of scheduling of subtasks:

---

Procedure 1

//  $\tau_{k,l}$  denotes the subtask that is released inside the busy period

Set  $t = 0$

Do

    Among subtasks with no predecessors select subtasks that arrived in the interval  $[t_B, t_B + t]$ ;

    Execute a subtask  $\tau_{k,l}$  with minimum deadline;  $t = t + c_{k,l}$ ;

    Remove the executed subtask,  $\tau_{k,l}$ , from its associated task graph;

Until processor becomes idle

---

## 4 Response Time Analysis of Sporadic Tasks

To find the WCRT of analyzed subtask, we must build the critical instant, leading to the calculation of the worst case execution interference. Since in our model each task consists of subtasks with arbitrary release times, we must take into account that the critical instance may not be obtained by occurrence of all tasks at the starting instance of the busy period, as was considered as the worst case in [1, 3]. Let us consider coincidence of which subtasks of a task  $\tau_k$  to  $t_B$  would result in the critical instance. To do so, first we shall categorize subtasks of the task  $\tau_k$  in two sets:  $\text{Set}_k^H$ : composed of subtasks with  $D_{k,l} \leq d$ ,  $\text{Set}_k^L$ : composed of subtasks with  $D_{k,l} > d$ . According to the definition of deadline— $d$  busy period, only the subtasks of task  $\tau_k$ ,  $k \neq a$ , included in  $\text{Set}_k^H$  would be considered to delay the execution of the subtask  $\tau_{a,b}$ , by arriving inside a deadline— $d$  busy period. Therefore, a deadline— $d$  busy period could be constructed by releasing one subtask of each task with  $D_{k,l} \leq d$  at  $t_B$ . However, we don't know which subtask coincidence to  $t_B$  corresponds to the critical instant for the analysis of the subtask  $\tau_{a,b}$ . So in the procedure of considering critical instance of subtask  $\tau_{a,b}$ , we must study all possible deadline— $d$  busy periods created by choosing one higher priority subtask in each task to occur at time  $t = t_B$ . Notice that, from [14] we have, the computation of the invocation of subtasks released before  $t_B$  can be ignored provided that an additional deadline— $d$  busy period could be examined, which starts at the release of that invocations.

Let us consider possible blocking times due to non-preemptability. Suppose that a subtask  $\tau_{p,q}$  with  $D_{p,q} > d$  starts its execution one unit time before releasing higher priority subtasks at  $t_B$ . Due to non-preemptability, the subtask  $\tau_{p,q}$  has to be executed completely before starting of the execution of higher priority subtasks. Consequently, execution of subtask  $\tau_{p,q}$  will allow the successors of  $\tau_{p,q}$ , denoted by  $\tau_{p,j}$ , with  $d_{p,j} \leq d$  to possibly be executed in the considered deadline— $d$  busy period. Furthermore, since each  $\tau_{p,q}$  could be followed by different amount of computation time of higher priority subtasks, we tend to consider all possible deadline— $d$  busy periods by further incurring all possible priority inversions.

Notice that, in this case, to take care of definition of the busy period, we move  $t_B$  to starting instance of execution of  $\tau_{p,q}$ . Consequently, all higher priority subtasks will be released at time  $t = t_B + 1$ .

**Theorem 1** *The WCRT of subtask  $\tau_{a,b}$  with release time at  $t_B \leq r < t_B + L$  and absolute deadline  $d$ , is found in one of the following cases:*

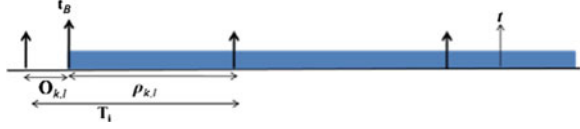
1. *In a deadline— $d$  busy period of subtask  $\tau_{a,b}$  where the first activation of some subtask  $\tau_{k,l}$  of task  $\tau_k$ ,  $k \neq a$ , with  $D_{k,l} \leq d$  coincides with  $t = t_B$  and task activations of  $\tau_k$  occur periodically with their maximum rate inside the busy period.*
2. *In a deadline — $d$  busy period of subtask  $\tau_{a,b}$  where the first activation of some subtask  $\tau_{k,l}$  of task  $\tau_k$ ,  $k \neq a$ , with  $D_{k,l} \leq d$  coincides with  $t = t_B + 1$  and task activations of  $\tau_k$  occur periodically with their maximum rate inside the busy period, also a priority inversion occurs at time  $t = t_B + 1$ , where one subtask  $\tau_{p,q}$ ,  $p \neq a$  with  $D_{p,q} > d$  activates at time  $t = t_B$ .*

*Proof* consider the first case: let  $t_1 > t_B$  be the instant at which a subtask  $\tau_{k,l}$ ,  $k \neq a$  with absolute deadline  $d_{k,l}$  is activated the first time inside the busy period, and let  $d_{k,l} \leq d$ . Suppose that we move  $t_1$  to coincide with  $t = t_B$  in this circumstance, it is possible that an activation of successor subtasks of subtask  $\tau_{k,l}$ , denoted by  $\tau_{k,j}$ , with an absolute deadline after instant  $d$  may have been moved earlier to have a deadline before or at  $d$ , thus it would possibly increase the response time of analyzed subtask. The proof of second case, is similar to above discussions.  $\square$

Based on Theorem 1, in the procedure of finding the critical instance of subtask  $\tau_{a,b}$  released at time  $t_B \leq r < t_B + L$ , we must study all possible deadline— $d$  busy periods created by choosing one higher priority subtask in each task to occur at time  $t = t_B$ . Further we must study all possible deadline— $d$  busy periods created by choosing one higher priority subtask in each task to occur at time  $t = t_B + 1$  where one lower priority subtask in one task occurs at time  $t = t_B$  and consequently incurs priority inversion at time  $t = t_B + 1$ . Given all possible deadline— $d$  busy periods, the largest response time of subtask  $\tau_{a,b}$  is accounted for its WCRT, denote by  $R_{i,j}$ . Without loss of generality, in the rest of this paper, we shall denote the time  $t = t_B + 1$ , in the second condition of Theorem 1, by  $t = t_B$ . Further the coincided subtask of task  $\tau_k$  to  $t_B$  is denoted by  $\tau_{k,l}$ . Through analysis expressed in Eqs. 2–10 we shall assume that task  $\tau_a$  consists of one subtask,  $\tau_{a,b}$ , only. In the latter part of this section we will extend the analysis to consider response time of  $\tau_{a,b}$  where the task  $\tau_a$  includes a sporadic graph with precedence constraints.

Let us first consider a complete activation of task  $\tau_k$  that occurs in the time interval  $[t_B, t]$  where  $t > t_B$ . This activation occurs before  $t$  so that  $t - \text{'activation time'} \geq T_k$  and  $\text{'activation time'} - t_B \geq 0$ . We denote the total number of complete activations of task  $\tau_k$  that occur in the time interval  $[t_B, t]$ , by  $N_{k,l}(t)$ . Remember that in our analysis we assume that the subtask  $\tau_{k,l}$  has been released

**Fig. 2** A scenario to obtain complete activations of task  $\tau_k$  up to time  $t$



at  $t_B$ . From Fig. 2, it is easy to see that  $N_{k,l}(t)$  could be obtained by:  $N_{k,l}(t) = \frac{t - \rho_{k,l}}{T_k}$ , where  $\rho_{k,l}$  represents the phase between  $t_B$  and the first activation of task  $\tau_k$  inside the busy period. Hence we have  $\rho_{k,l} = T - O_{k,l}$  if  $O_{k,l} > 0$  and  $\rho_{k,l} = 0$  otherwise. The total time required to schedule activations of task  $\tau_k$  occurred completely in the time interval  $[t_B, t]$  equals:

$$C_{k,l}(t) = \left\lfloor \frac{t - \rho_{k,l}}{T_k} \right\rfloor \cdot \sum_{i=1}^{n_k} c_{k,i} \quad (2)$$

Figure 3 represents a scenario for the alignment of task  $\tau_k$  arrival pattern after  $t_B$ . Upper part of this figure corresponds to the activation and deadline of subtask  $\tau_{a,b}$ , and the lower part represents the activation of subtasks of task  $\tau_k$ . In this figure,  $N_{k,2}(t') = 0$ , because  $t' - t_2 < T_k$ . Further, since  $t'' - t_2 > T_k$ ,  $t_2 - t_B > 0$  also  $t'' - t_3 > T_k$ ,  $t_3 - t_B > 0$ , we have  $N_{k,2}(t'') = 2$ .

Notice that in above discussion we only find the time required to execute  $N_{k,l}(t)$  activations of task  $\tau_k$ . However, among those, activations of task  $\tau_k$  released after  $d - D_k$  would not contribute to the response time of analyzed subtask. Thus at each given time interval  $[t_B, t]$ , only activations of task  $\tau_k$  with deadlines before or at  $d$ , can contribute to the response time of analyzed subtask. That is at each given time interval  $[t_B, t]$ ,  $C_{k,l}(t)$  is bounded by:

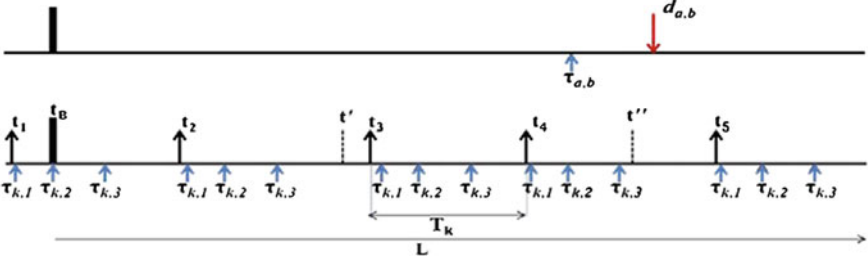
$$G_{k,l}(d) = \left( 1 + \left\lfloor \frac{d - (D_k + \varphi_{k,l})}{T_k} \right\rfloor \right) \cdot \sum_{i=1}^{n_k} c_{k,i} \quad (3)$$

Therefore, the contribution of  $N_{k,l}(t)$  complete activations of task  $\tau_k$  to the response time of analyzed subtask, at each given time interval  $[t_B, t]$ , is given by:

$$W_{k,l}(t, d) = \min(C_{k,l}(t), G_{k,l}(d)) \quad (4)$$

To see this, redo example in Fig. 3, let us assume that  $D_{k,1} = D_{k,3} = T_k$ ,  $D_{k,2} = 3 \cdot T_k - O_{k,2}$  and hence  $D_k = 3 \cdot T_k$ . In this case, we have  $C_{k,2}(t'') = 2 \cdot \sum_{i=1}^3 c_{k,i}$ ,  $G_{k,2}(d) = 0$  and consequently  $W_{k,l}(t'', d) = 0$ .

Equation 4 only obtains contribution of complete activations of task  $\tau_k$  that occur inside the time interval  $[t_B, t]$ . The problem that remains is to obtain contribution of activations that do not interfere to the response time of subtask under analysis, completely. We shall categorize those contributions of task  $\tau_k$  in two sets. Set A: Activations included in this set occur completely inside the time interval  $[t_B, t]$  but they have deadline greater than  $d$ . Thus those activations do not interfere to the response time of  $\tau_{a,b}$ , completely. For example in Fig. 3; consider



**Fig. 3** An example for calculating required time to schedule task  $\tau_k$  up to time  $t$

$D_{k,1} = D_{k,3} = T_k$ ,  $D_{k,2} = 3 \cdot T_k - O_{k,2}$ . In this case, we have  $W_{k,2}(t', d) = 0$ , but activations of task  $\tau_k$  occurred at times  $t_2, t_3$  consist of some subtasks that would delay the response time of  $\tau_{a,b}$ ; subtasks  $\tau_{k,1}^2, \tau_{k,1}^3$ . We shall define  $\delta_{k,l}^A(t, d)$  to obtain total computation time of subtasks of activations of task  $\tau_k$  in Set A by:

$$\delta_{k,l}^A(t, d) = \max(E_k) \quad (5)$$

where  $\max(E_k)$  represents summation of computation time of subtasks of activations in Set A that have an absolute deadline less than or equal to  $d$  and activation time before or at  $t$ , so that all their predecessors also have an absolute deadline less than or equal to  $d$  and activation time before or at  $t$ . In Fig. 3, consider  $D_{k,1} = D_{k,3} = T_k$ ,  $D_{k,2} = 3 \cdot T_k - O_{k,2}$ , we have  $\delta_{k,l}^A(t, d) = 2 \cdot c_{k,1}$ . Set B: Activations included in this set do not occur inside the time interval  $[t_B, t]$ , completely. This set may consists of two members only. The first one is the first activation that occurs immediately before  $t_B$ , where  $O_{k,l} \neq 0$ ; e.g. activation of task  $\tau_k$  occurred at time  $t_1$ , in Fig. 3. The second member of Set B is the last activation, where  $t - t' \text{ activationtime}' < T_k$ . Consider time interval  $[t_B, t]$  in Fig. 3, activation of task  $\tau_k$  occurs at time  $t_2$ , corresponds to this element. We shall define  $\delta_{k,l}^B(t, d)$  to account for total computation time of subtasks of activations of task  $\tau_k$  in Set B by:

$$\delta_{k,l}^B(t, d) = \max(E_k) \quad (6)$$

where  $\max(E_k)$  is defined as for Eq. 5. Finally, we shall define  $\delta_{k,l}(t, d)$  to obtain total computation time of subtasks of activations of task  $\tau_k$  in Set A and Set B by:

$$\delta_{k,l}(t, d) = \delta_{k,l}^A(t, d) + \delta_{k,l}^B(t, d) \quad (7)$$

So far, we have studied the contribution of task  $\tau_k$  to the response time of  $\tau_{a,b}$  that has been released until time  $t$ . In general the study of response time of a subtask  $\tau_{a,b}$  with deadline  $d$  is an iterative procedure. The basic idea is that in each step the obtained busy period, must be the end of execution of all instances of all tasks with deadlines less than or equal to  $d$ , that have been released in previous steps. Toward this we shall define  $LN_{a,b}(t, d)^{(k)}$  to represent the length of the

resulting busy period in the  $k$ 'th iteration of response time analysis of subtask  $\tau_{a,b}$  with deadline  $d$ , where  $t$  is substituted with the length of obtained busy period in previous step.  $LN_{a,b}(t, d)^{(k)}$  is obtained by the iterative equation Eq. 9. This equation represents the iterative procedure where at each step the length of busy period is given by summation of execution times obtained in Eqs. 4, 7. We shall initiate this iterative procedure by Eq. 8. In Eq. 8,  $c_{k,l}$  represents the execution time of coincided subtask of each task to  $t_B$ . The upper part of Eq. 8 indicates the initial value of  $LN_{a,b}(t, d)$  in a scenario constructed based on first term of Theorem 1. The lower part of Eq. 8 indicates a scenario constructed based on second condition of Theorem 1. The iteration is halted where the computations converge.

$$LN_{a,b}(t, d)^{(0)} = \begin{cases} \sum_{\forall k} c_{k,l} & \text{first scenario in Theorem. 1} \\ \sum_{\forall k} c_{k,l} - 1 & \text{second scenario in Theorem. 1} \end{cases} \quad (8)$$

$$LN_{a,b}(t, d)^{(m+1)} = \sum_k W_{k,l}(LN_{a,b}(t, d)^{(m)}, d) + \sum_k \delta_{k,l}(LN_{a,b}(t, d)^{(m)}, d) \quad (9)$$

The response time of subtask  $\tau_{a,b}$  where it occurs at time  $t_B \leq r < t_B + L$  is:

$$R_{a,b}(r) = LN_{a,b}(t, d) + c_{a,b} - r \quad (10)$$

Then the worst case response time of subtask  $\tau_{a,b}$  can be obtained by Eq. 1.

The problem that remains to be solved is to determine the response time of subtask  $\tau_{a,b}$  where task  $\tau_a$  includes a sporadic graph with precedence constraints. Consider a scenario where  $\tau_{a,b}$  is released at  $r$ . In this case, the response time of  $\tau_{a,b}$  is influenced by the order of execution of all subtasks of task  $\tau_a$  that must be scheduled before  $\tau_{a,b}$ . Further, this sequence of subtasks of task  $\tau_a$  may not be the same as sequence of subtasks in another scenario where  $\tau_{a,b}$  is released at  $r'$ . Therefore, it is of great importance to determine the order of subtasks of task  $\tau_a$  that must be scheduled before  $\tau_{a,b}$  for each release time of  $\tau_{a,b}$ .

Let us denote the current subtask of task  $\tau_a$  that its response time has been considered by  $\tau_{a,q}$  and the next chosen subtask by  $\tau_{a,p}$ . Consider a scenario where  $\tau_{a,b}$  is released at  $r$ . In order to find the correct response time of subtask  $\tau_{a,b}$ , we shall use Procedure 1, to determine the correct sequence of subtasks of task  $\tau_a$  that must be scheduled before  $\tau_{a,b}$ . This procedure would let us to consider the correct sequence of subtasks of task  $\tau_a$  that must be scheduled before  $\tau_{a,b}$  and consequently the correct response time of subtask  $\tau_{a,b}$ . Because  $\tau_{a,q}$  and all the subtask of task  $\tau_a$  that has been considered by Eqs. 8 or 9 will be dropped from the sporadic graph of task  $\tau_a$ . Finally,  $\tau_{a,p}$  would be a subtask with minimum deadline among subtasks of task  $\tau_a$  with no predecessor and arrival time at or before completion of execution of  $\tau_{a,q}$ .



### 5 Comparison with Existing Techniques

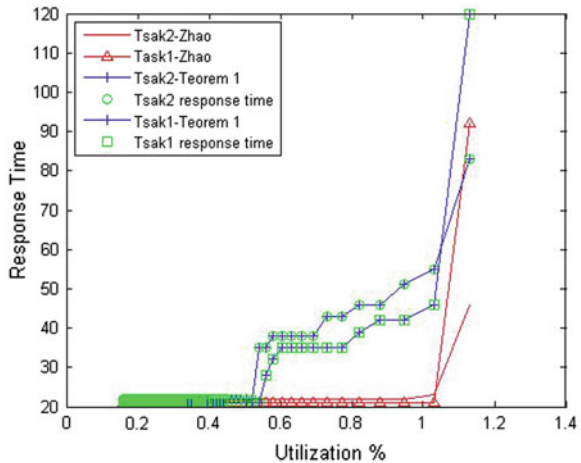
We have compared the results of our proposed analysis, with the results obtained by [3], the only bibliography of our work. Reference [3] defines sporadic task model with a major constraint; simultaneous arrival time of all subtasks of a task after the arrival of the external event. Due to space constraint we refer reader to [3] for more details. We have conducted a set of simulations with a set of tasks represented in Table 1.

Figure 4 compares the WCRT of tasks  $\tau_1$ ,  $\tau_2$  with WCRT obtained by Zhao’s technique and the response times obtained using our approach. The X axis represents processor utilization and the Y axis represents WCRT. The processor utilization is raised by changing the period of task under analysis in the interval [9,300], where at 9 the system no longer meets its deadlines and at 300 the utilization is close to zero. It can be seen that for utilization values of 50 % and lower, our approach and the algorithm in [3] obtain same results. However, for utilization values of 50 % and higher, increasing the utilization of the task cannot increase the obtained WCRT by Zhao’s approach but we can get an increase of response time over two times greater than Zhao’s algorithm. This result is due to graph to canonical chain transformation in Zhao’s approach where it incurs deadlines of predecessor subtasks of the subtask under analysis to be increased. Specifically, in a case where the analyzed subtask has a larger deadline than its

**Table 1** Tasks parameters

	$\tau_{1,1}$	$\tau_{1,2}$	$\tau_{1,3}$	$\tau_{2,1}$	$\tau_{2,2}$	$\tau_{2,3}$	$\tau_{3,1}$	$\tau_{3,2}$
$C_{i,j}$	4	3	1	5	3	1	2	3
$D_{i,j}$	10	12	50	8	15	55	50	18
$T_i$	100			100			100	

**Fig. 4** Response time comparison



successors and the associated task has a tight period, Zhao analysis would imply that the system is still schedulable but in fact tasks will miss their deadlines in the worst case.

## 6 Conclusion

In this paper we studied WCRT analysis of sporadic tasks scheduled under non-preemptive EDF scheduling. The objective is to obtain precise WCRTs of a task with arbitrary timing requirements that is not considered in previous works. A nice feature of our work is that it exploits precedence constraints between subtasks in an accurate way while taking advantage of deadline of subtasks. We find via simulation results that our methodology provides accurate results compare with existing work in [3].

**Acknowledgments** This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2010-0024245).

## References

1. Spuri, M.: Analysis of deadline scheduled real-time systems. Research report 2772, INRIA, France, Jan. 1996
2. Zhao, H.X., Midonnet, S., George, L.: Worst case response time analysis of sporadic task graphs with fixed priority scheduling on a uniprocessor. In: 11th IEEE RTCSA, HongKong, pp 23–29 (2005)
3. Zhao, H., George, L., Midonnet, S.: Worst case response time analysis of sporadic task graphs with EDF non-preemptive Scheduling on a uniprocessor. In: Proceedings of the third IEEE International Symposium on Computer and Communication (ICAS'07) 2007
4. Liu, C.L., Layland, J.W.: Scheduling algorithms for multiprogramming in a hard real-time environment. *J ACM* **20**(1), 40–61 (1973)
5. Mangeruca, L., Baleani, M., Ferrari, A., Sangiovanni-Vincentelli.: Uniprocessor scheduling under precedence constraints for embedded systems design. In: *ACM Transaction on Embedded Computing System* (2007)
6. Buttazzo, G., Bini E., Wu, Y.: Partitioning parallel applications on a multiprocessor reservations. In: Proceedings. of the 22nd Euromicro Conference on Real-Time Systems (ECRTS 10), Brussels, Belgium, July 6-9, 2010
7. Spuri, M.: Holistic analysis of deadline scheduled real-time distributed systems, RR-2873. INRIA, France (1996)
8. Pellizzoni, R., Lipari, G.: Improved schedulability analysis of real-time transactions with earliest deadline scheduling. Technical report RETIS-TR-2005-01, Scuola superiore sant'anna 2004
9. Palencia, J.C., Harbour, M.G.: Offset-based response time analysis of distributed systems scheduled under EDF. In: 15th Euromicro Conference on Real-Time Systems (ECRTS'03) (2003)

10. Palencia, J.C., Harbour, M. G.: Exploiting precedence relations in the schedulability analysis of distributed real-time systems. In: The 20th IEEE Real-Time Systems Symposium, pp. 328–339 1999
11. Palencia, J.C., Harbour, M. G.: Schedulability analysis for tasks with static and dynamic offsets. In: Proceedings of the 19th IEEE Real-Time Systems Symposium 1998
12. Jayachandran, P., Abdelzaher, T.: End-To-End delay analysis of distributed systems with cycles in the task graph. In: 21st Euromicro Conference on Real-Time Systems (2009)
13. Redell, O.: Analysis of tree-shaped transactions in distributed real time systems. In: Proceedings of the 12th 16th Euromicro Conference on Real-Time Systems (ECRTS'04) (2004)
14. Tindell, K.: Adding time-offsets to schedulability analysis, technical report YCS 221. Department of computer science, University of York, England 1994