ICIC Express Letters
Part B: Applications
Volume **10**, Number **1**, January **2019**

 ISSN 2185-2766
pp. **17–24**

PERFORMANCE EVALUATION OF NON-PREEMPTIVE EARLIEST DEADLINE FIRST SCHEDULING TECHNIQUES

Jamal Alhiyafi

Department of Computer Science
College of Computer Science and Information Technology
Imam Abdulrahman Bin Faisal University
P.O. Box 1982, Dammam 31441, Saudi Arabia
jalhiyafi@iau.edu.sa

Received August 2018; accepted November 2018

Abstract. *The deadlines associated with periodic tasks are of great importance to the functionality of a real-time system. To fulfill the deadline requirement of real-time systems, various scheduling policies are applied. Among these scheduling algorithms, the earliest deadline first scheduling algorithm is an optimal technique under preemptive class which promises 100% system utilization. Unfortunately, this approach does not exhibit optimal performance when considered for non-preemptive systems. Due to its wider acceptance in industry, the earliest deadline first scheduling algorithm has been studied extensively as an optimal scheduling algorithm and several feasibility techniques do exist to validate the timing constraints of the system. In this work, we extend our previous results and study the behavior of closely related feasibility tests under varying system utilizations for the non-preemptive earliest deadline first algorithm and show that the performance of a non-preemptive scheduling algorithm degrades significantly at a system utilization of 80% or higher, irrespective of the scheduling policy.*
Keywords: Operating systems, Non-preemptive systems, Online schedulability, Real-time systems

1. **Introduction.** The main goal of general operating systems is resource management and resources such as CPU need to be managed efficiently to maximize overall system performance. In real-time operating systems, in addition to efficient resource management, another constraint is meeting the deadlines associated with task and hence theatrical validation is needed. The operating systems scheduler is responsible for scheduling tasks on the CPU and provides two main classes of schedulers, namely preemptive and non-preemptive. Preemptive scheduling results in higher resource utilization but the associated overhead for context switch is high. On the other hand, non-preemptive scheduling provides reduced system utilization but offers the advantage of simple implementation.

Respecting the task deadline is one of the most critical attributes of real-time systems and cannot be compromised under any possible scenario. In such systems, heuristic approaches can result in improved resource utilization but such approaches become irrelevant in real-time systems, especially for hard real-time systems where deadliness can result in catastrophic consequences. To validate the timing constraint of real-time systems, statistical analysis becomes irrelevant and such a system must be validated mathematically.

Scheduling of a real-time system is mainly divided into two broader classes: (i) preemptive, and (ii) non-preemptive. Preemptive scheduling allows task interruption at any point in time [1-12], while non-preemptive algorithms allow the running task to execute continuously till completion of task CPU requirement [13-16]. In comparison to preemptive scheduling algorithms, non-preemptive algorithms are simple to implement due to a less number of context switching requirements [13].

DOI: 10.24507/icicelb.10.01.17

For preemptive systems, Earliest Deadline First (EDF) scheduling is an optimal technique [17] established in 1973 for a set of periodic tasks with relative deadlines equal to their task periods. However, EDF loses its importance for a non-preemptive counterpart as the system allows a running task till completion. The available feasibility techniques [13,14,16,18,19] have high computational complexity associated and hence becomes unuseable in systems, especially when deployed to online systems. In such systems, more time is wasted calculating the feasibility of the task rather than using CPU for real processing of tasks. This issue is of paramount importance for online schedulability of the system. Baruah and Chakraborty in [20] showed that polynomial time approximation algorithms exist for both preemptive and non-preemptive scheduling classes of recurring task models. Authors in [13] established a feasibility test for non-preemptive EDF that was further extended in [14].

Under dynamic scheduling, the non-preemptive scheduling has been utilized in many commercial operating systems such as Micro-Controller Operating Systems (uC/OS) [21], Real Time Operating System (RTOS) [22], and Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP) [23]. However, the computation cost associated with respective feasibility condition is very high. Though promising, the complexity of such systems is still pseudo-polynomial and there exist a plethora of results. To the best of our knowledge, no analysis of such systems has been performed till date. In this work, we compare the existing techniques to assist the designer in making informed decisions while designing the system of their interest.

For non-preemptive execution, authors in [24] introduced a precautious-rate monotonic as a predictable linear-time. The test in [24] is applicable to online non-preemptive scheduling systems. Authors in [15] solved the open problem of non-preemptive scheduling of mixed-criticality systems. Under the work done in [15], a solution for harmonic task set on a uni-processor system was presented. The problem of testing feasibility for all tasks was formulated into a sufficient condition in [25]; however, the complexity associated with such conditions is still high. In this work, we extend our previous work in [25] and study the performance of various feasibility tests for the task sets under various sizes and utilizations. However, experimental results are missing in [25] and there is a need to evaluate related approaches for various system utilization. We highlight the pros and cons of existing techniques and show that technique presented in [25] is the most efficient one. Our analysis shows that the performance on preemptive EDF degrades significantly with higher CPU utilization as it is very likely that the system is executing a low priority task at any point in time and cannot preempt in favor of awaiting higher priority tasks due to the restriction of non-preemption of running tasks.

The rest of the paper is divided into five sections. Section 2 discusses the related work that provides the foundation for our work and sketch of the system model. The main results are analyzed in Section 3. Experimental results are discussed in Section 4, and the paper is concluded in Section 5.

2. **Background and Related Work.**

In real-time systems, system predictably is achieved by allocating respective priorities to individual tasks in the system. These priorities are subject to some pre-established criteria. The most popular one is the activation rate or deadline [26,27]. Under a single processor system, the fixed-priority scheduling algorithm assigns the same priority to all of the jobs/instances of a particular task. A task then creates a sequence on jobs after regular intervals called time periods [26,27]. Unlike fixed priority systems, dynamic-priority scheduling algorithms place no restrictions on priority assignment. In dynamic-priority algorithms, when two tasks have the same absolute deadlines, the algorithm selects the task at random. For this work, in our task model, task periods of any two tasks cannot be the same; hence, a unique priority is assigned to each task.

Assuming an unlimited number of priority levels, a necessary and sufficient condition saying that the system utilization must be less than or equal to 100% for EDF was derived in [17]. The test established in [17] is a simple sufficient condition having $O(n)$ complexity and most suitable for online schedulability analysis. This formulation allows EDF to suppress the fixed-priority counterpart such as Rate-Monotonic (RM) algorithm from a utilization perspective. The sufficient condition derived for RM also has $O(n)$ complexity but it traded system utilization i.e., the test can guaranty system feasibility when total utilization is not more than $ln(2)$ which results in wasting 31% of CPU slots.

In 1991, the most celebrated results were derived in [13] for non-preemptive EDF. The work done in [13] was extended later in [14,16]. However, all of these tests shared a high computational complexity which becomes impractical for online systems where feasibility should be determined efficiently.

Our system model consists of an infinite number of jobs and each job belongs to a task. The system has n number of tasks to be executed on a uni-processor system. In the periodic task model, a task T_i is described by three attributes, task period p_i that is the time between any two consecutive instances of T_i, an execution time c_i that a task T_i must get before its next occurrence, and a relative deadline d_i. For each task, the system utilization is shown by c_i/p_i. The cumulative demand for n tasks is represented by $\sum_{i=0}^{n}(c_i/p_i)$.

Our non-concrete task set is represented by $T = \{T_1, T_2, \ldots, T_n\}$, where an instance of T_i is generated at every integer multiple of p_i: the k^{th} instance of T_i occurs at time $t_k = t_{k-1} + p_i$. We show the results for an implicit deadline model of non-concrete [1] for a real-time system. In our system, interrupting a running task has been disabled and the task is executed till completion.

Results have been previously established theoretically, but there is a lack of experimental analysis for non-preemptive class of dynamic priority systems for periodic tasks. To the best of our knowledge, no experimentation on evaluation of non-preemptive dynamic priority scheduling algorithms has been done prior to this work. We fulfill this gap by highlighting the advantages and disadvantages of related techniques and provide experimental results analyzing the behavior of non-preemptive periodic systems under system utilization starting from 70% to 100% with a step size of 10%. The work described in this paper will provide insight in regard to feasibility tests for dynamic class and provide a clear comparison of two alternative solutions.

3. **Feasibility Analysis of Non-Preemptive Tasks.** System utilization of more than 100% is impossible via a single processor with EDF even under preemptive scheduling [17]. This condition is also valid in non-preemptive cases. Under a preemptive case, the entire task set is always feasible on uni-processor systems as long as the system utilization is

$$\sum_{i=0}^{n} \frac{c_i}{p_i} \leq 1. \tag{1}$$

There are cases when the task set that is schedulable using preemptive scheduling can become un-schedulable on a uni-processor system when non-preemptive scheduling is used. Thus, a simple analysis test like Inequality (1) is sufficient for a preemptive case. The non-preemptive counterpart, however, is more complicated as far as the feasibility of the entire task set is concerned. Non-preemptive scheduling allows the running task to continue till completion, while a higher priority task has to wait. Various extensions on no preemptive scheduling have been proposed by assigning a quantum to tasks; however, the running task has to leave the CPU once the quantum expires. In [13], the feasibility problem of EDF under a non-preemptive case was answered with Theorem 3.1.

Theorem 3.1. ([13]) *A periodic task set that is sorted in increasing order of the task period can only be scheduled feasibly under a non-preemptive EDF scheduling policy iff,*

$$\sum_{i=0}^{n} \frac{c_i}{p_i} \leq 1$$

$$\forall i, (1 < i \leq n),\ \forall t, (p_1 \leq t \leq p_i) : c_i + \sum_{k=1}^{i-1} \left\lfloor \frac{t}{p_k} \right\rfloor c_k \leq t. \tag{2}$$

From Theorem 3.1, it is clear that the system must evaluate t in a time interval of $(p_1 < t < p_i)$ to determine task schedulability. The result of Theorem 3.1 was later simplified in [19] to confine t to set of a few scheduling points. The work done in [14] can be reworded as:

Theorem 3.2. ([14]) *A periodic task set that is sorted in increasing order of the task period can only be scheduled feasibly under a non-preemptive EDF scheduling policy, at a constant slowdown factor of η, if*

$$\frac{1}{\eta} \sum_{i=0}^{n} \frac{c_i}{p_i} \leq 1 \tag{3}$$

$$\forall i, (1 < i \leq n),\ \forall t, (p_1 \leq t \leq p_i) : \frac{1}{\eta} \left(c_i + \sum_{k=1}^{i-1} \left\lfloor \frac{t}{p_k} \right\rfloor c_k \right) \leq t, \tag{4}$$

$$\text{where } t \in S_i = \left\{ lp_i;\ \forall j, (j = 1, \ldots, i);\ \forall l, \left(l = 1, \ldots, \left\lfloor \frac{p_i}{p_j} \right\rfloor \right) \right\}$$

where S_i is called a set of scheduling points for T_i. The set of scheduling points plays an important role in schedulability analysis of non-preemptive systems. Recently, another sufficient condition was established in [25] with the intention to lower the computation cost of the work done in 3.2. This technique can answer feasibility of the entire task set by only checking the schedulability of the lowest task as this task will only get CPU time when there is no pending higher priority task in the queue. The test suggests:

Theorem 3.3. ([25]) *A periodic task set that is sorted in increasing order of the task period can only be scheduled feasibly under a non-preemptive EDF scheduling policy if*

$$\sum_{i=0}^{n} \frac{c_i}{p_i} \leq 1$$

$$\forall t, (t \in S_n) : c_n + \sum_{k=1}^{n-1} \left\lfloor \frac{t}{p_k} \right\rfloor c_k \leq t. \tag{5}$$

It is worth noting that Theorem 3.1 is both a necessary and sufficient condition while Theorems 3.2 and 3.3 are sufficient conditions only. It can be seen that the number of scheduling points is fixed in Theorem 3.3 and thus, feasibility is tested for any $t \in S_i$ in a task T_i. All of the above results are of pseudo-polynomial nature with closed and open interval of points. In this work, we discuss the efficiency of computation cost of closed interval tests and count the number of scheduling points needed for both Theorems 3.2 and 3.3. The computational cost has been reduced significantly with Inequality (5) as only set S_n needs to be tested for task T_n which eventually leads us to the conclusion that the task set is feasible on a uni-processor system. In case Inequality (5) does not hold, the entire system is declared infeasible. Theorem 3.3 has been established theoretically in [25] while the experiential evaluation is performed in this work. It is also evident that for a feasible set, Theorem 3.2 will need to analyze all points in sets, i.e., $S_1, S_2, \ldots, S_n$, while Theorem 3.3 only needs the points in one set generated by the lowest priority task.

Having the aforementioned results, we compare the related techniques in the following section.

4. **Experimental Evaluation.** For experimentation, we generated a task set from size 10 to 100 with step size of 10. Task periods were generated with uniform distribution and the task computation demand for an individual task was obtained in the range of 1 to task period. To have confidence in our results, each experiment ran 200 times and the average values plotted accordingly. For the purpose of analysis, we represent Theorem 3.1 by Improved Test (IT) and Theorem 3.3 by Enhanced Test (ET). We compared the run time performance on IT and ET with system utilizations of 0.70, 0.80, 0.90 and 1.0. We drew random values for a fair comparison of both solutions. As it is understood that a task set is normally unschedulable with utilization higher than 0.9. The same has been reflected in our experimental analysis.

It can be seen in Figures 1-4 that both IT and ET take less time when the task set is low and take a longer time when the task size increases. This behavior is due to the fact that only a few calculations are needed for the smaller sized task set. On the other hand, the number of calculations increases significantly for larger task sets, as the task period becomes larger. In such a scenario, p_i/p_1 inequalities need to be tested for each task which is the worst case for the IT technique. In contrast, ET performs better due to testing only p_n/p_1 points in total for the entire task set. In Figure 1, maximum tasks are non-preemptive schedulable at 70% system utilization. The same trend is shown in Figure 2, where the CPU requirement is 80%. This is the point where larger task set becomes infeasible due to more interference of lower priority tasks on execution of the higher priority tasks. At higher utilization, the system infeasibility is concluded earlier as reflected in Figure 3 and Figure 4. It can be seen that both IT and ET tests take more time as maximum tasks need to be evaluated before infeasibility is determined. However, the infeasibility of the overall system is determined much early with few tasks analyzed when system utilization is 100%, as shown in Figure 4. In all cases (Figures 1-4), ET outperforms IT due to its implicit characteristic of confining scheduling points to a subset S_n only. Our experimental results are aligned with the theoretical formulation of ET, which is the intended contribution of this work.

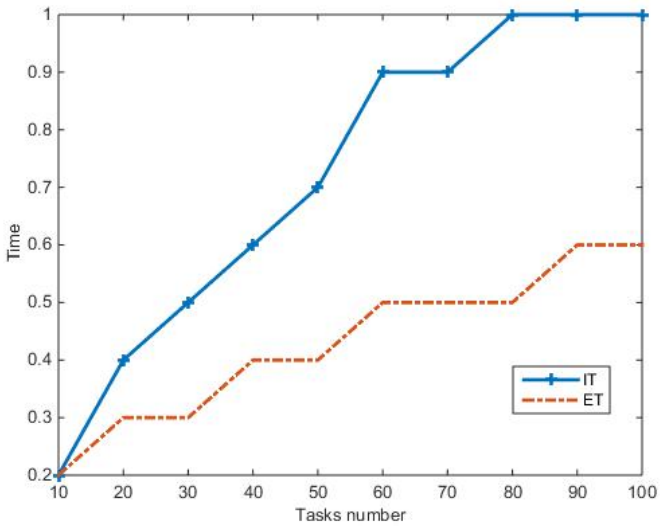

FIGURE 1. Performance at 70 percent system utilization

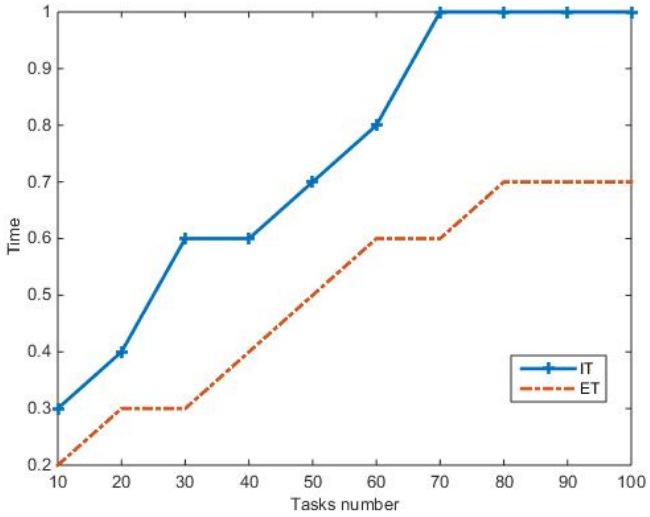

FIGURE 2. Performance at 80 percent system utilization

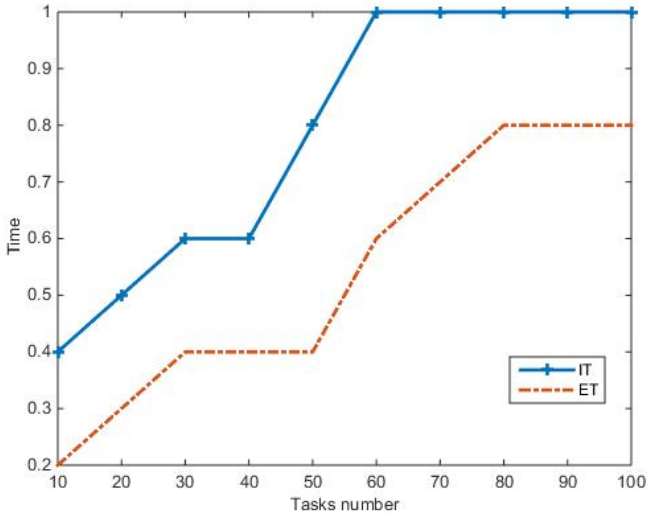

FIGURE 3. Performance at 90 percent system utilization

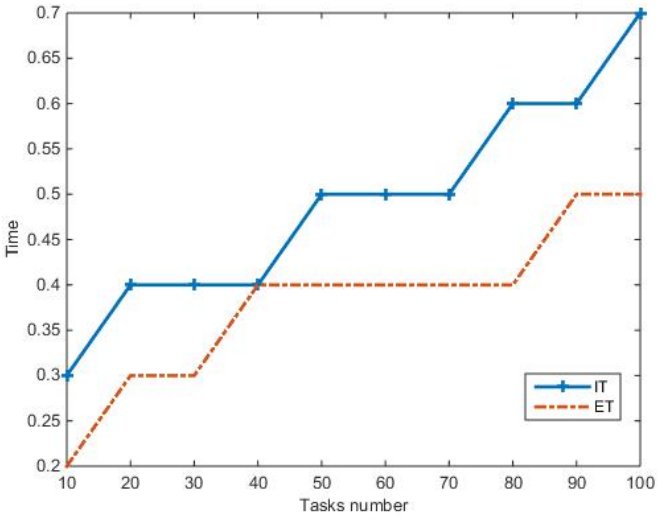

FIGURE 4. Performance at 100 percent system utilization

5. **Conclusion.** In this work, we have studied the effect of non-preemptive scheduling on system feasibility. All related tests have been applied to a periodic task model where tasks are independent, and the number of priority levels is infinite. The task sets were analyzed under a dynamic priority assignment policy for non-preemptive cases. We drew the analysis for the task sets in sizes ranging from 10 to 100 at various CPU utilizations. It is concluded that the system utilization as well as the size of the task set has a significant impact on the feasibility of task due to the non-preemptive portion of the low priority tasks. As a future work, there is a need to study the behavior of non-preemptive feasibility tests by using a lowest priority first approach where feasibility of a system starts with the lowest priority.

REFERENCES

[1] L. George, N. Riverre and M. Spuri, Preemptive and non-preemptive real-time uniprocessor scheduling, *Research Report 2966*, INRIA, France, 1996.

[2] L. Sha, T. Abdelzaher, K.-E. Årzén, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky and A. K. Mok, Real-time scheduling theory: A historical perspective, *Real-Time Systems*, vol.28, no.2, pp.101-155, 2004.

[3] R. I. Davis and A. Burns, A survey of hard real-time scheduling for multiprocessor systems, *ACM Computing Surveys*, vol.43, no.4, pp.1-44, 2011.

[4] A. Burns, R. I. Davis, P. Wang and F. Zhang, Partitioned EDF scheduling for multiprocessors using a C=D Scheme, *Real-Time Systems*, vol.48, no.1, pp.3-33, 2012.

[5] N. Min-Allah, S. U. Khan, N. Ghani, J. Li, L. Wang and P. Bouvry, A comparative study of rate monotonic schedulability tests, *Journal of Supercomputing*, vol.59, no.3, pp.1419-1430, 2012.

[6] N. C. Audsley, A. Burns, R. I. Davis, K. W. Tindell and A. J. Wellings, Real-time system scheduling, in *Predictably Dependable Computing Systems*, B. Randell, J.-C. Laprie, H. Kopetz and B. Littlewood (eds.), ESPRIT Basic Research Series, Springer, 1995.

[7] N. Min-Allah, I. Ali, J. Xing and Y. Wang, Utilization bound for periodic task set with composite deadline, *Journal of Computers and Electrical Engineering*, vol.36, pp.1101-1109, 2010.

[8] N. Min-Allah, H. Hussain, S. U. Khan and A. Y. Zomaya, Power efficient rate monotonic scheduling for multi-core systems, *Journal of Parallel and Distributed Computing*, vol.72, no.1, pp.48-57, 2012.

[9] N. Min-Allah, S. U. Khan and Y. Wang, Optimal task execution times for periodic tasks using nonlinear constrained optimization, *Journal of Supercomputing*, vol.59, no.3, pp.1120-1138, 2012.

[10] S. Baruah, S. Funk and J. Goossens, Robustness results concerning EDF scheduling upon uniform multiprocessors, *IEEE Trans. Computers*, vol.52, no.9, pp.1185-1195, 2003.

[11] E. Bini and G. C. Buttazzo, The space of EDF deadlines: The exact region and a convex approximation, *Real-Time Systems*, vol.41, no.1, pp.27-51, 2009.

[12] N. Min-Allah and S. U. Khan, A hybrid test for faster feasibility analysis of periodic tasks, *International Journal of Innovative Computing, Information and Control*, vol.7, no.10, pp.5689-5698, 2011.

[13] K. Jeffay, D. F. Stanat and C. U. Martel, On non-preemptive scheduling of periodic and sporadic tasks, *Proc. of the Real-Time Systems Symposium*, pp.129-139, 1991.

[14] R. Jejurikar and R. Gupta, Energy aware non-preemptive scheduling for hard real-time systems, *Proc. of the 17th of Euromicro Conference on Real-Time Systems*, pp.21-30, 2005.

[15] M. Nasri and F. Gerhard, Open problems on non-preemptive scheduling of mixed-criticality real-time systems, *International Real-Time Scheduling Open Problems Seminar (RTSOPS'15)*, pp.17-18, 2015.

[16] S. Kim, J. Lee and J. Kim, Runtime feasibility check for non-preemptive real-time periodic tasks, *Information Processing Letters*, vol.97, no.3, pp.83-87, 2006.

[17] C. L. Liu and J. W. Layland, Scheduling algorithms for multiprogramming in a hard real-time environment, *Journal of the ACM*, vol.20, no.1, pp.40-61, 1973.

[18] S. Alrashed, A feasibility study of sorted tasks set, *ICIC Express Letters, Part B: Applications*, vol.9, no.9, pp.907-915, 2018.

[19] C.-Y. Chen, R.-C. Wu, C.-L. Pan, C.-M. Hsu and S.-H. Chen, Embedded controller design for curtain motor operation with bluetooth and modbus communication, *ICIC Express Letters, Part B: Applications*, vol.8, no.9, pp.1243-1252, 2017.

[20] S. K. Baruah and S. Chakraborty, Schedulability analysis of non-preemptive recurring real-time tasks, *Proc. of the 20th IEEE International Parallel & Distributed Processing Symposium*, Rhodes Island, Greece, 2006.

[21] *uC/OS*, https://www.micrium.com/rtos/, 2016.
[22] *FreeRTOS*, http://www.freertos.org/, 2016.
[23] A. G. Wilson and H. Nickisch, *Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP)*, Carnegie Mellon University, 2016.
[24] M. Nasri and M. Kargahi, Precautious-RM: A predictable non-preemptive scheduling algorithm for harmonic tasks, *Real-Time Systems*, vol.50, no.4, pp.548-584, 2014.
[25] S. Alrashed, J. Alhiyafi, A. Shafi and N. Min-Allah, An efficient schedulability condition for non-preemptive real-time systems at common scheduling points, *The Journal of Supercomputing*, vol.72, no.12, pp.4651-4661, 2016.
[26] J. W. S. Liu, *Real Time Systems*, Prentice Hall, 2000.
[27] C. M. Krishna and K. G Shin, *Real-Time Systems*, McGraw-Hill, 1997.