International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 8 (2018) pp. 5809-5814

Higher Initial Value for Time Demand Analysis

Saleh Alrashed

Imam Abdulrahman Bin Faisal University, Dammam, Saudi Arabia.

Abstract

Time demand analysis of real-time has been an active research area and a plethora of results have been derived for the classing time demand analysis by relaxing the assumption such as making task periods harmonic, restricting the scheduling points to a subset, obtain guess values, and avoiding testing system feasibly at unnecessary points. However, till date, the complexity of the time demand analysis still remains pseudo-polynomial. In this work, we propose a higher initial value for testing the feasibility of a lower priority task based on the feasibility analysis of immediate higher priority task. The proposed technique shows improvement over closely related counterparts. Experimental results are aligned with theoretical formulations presented in this paper.

Keywords: Operating Systems, Time Demand Analysis Real-Time Systems, Fixed-Priority Scheduling, Feasibility Analysis, Online Schedulability Test.

INTRODUCTION

A major challenge in the design of real-time embedded system is to validate its correctness. Such systems are not only expected to be logically correct but the timing constraints of these systems must also be respected. Various scheduling techniques have been proposed in literature to verify the correctness of real-time system. The real-time scheduling algorithms can be divided into two main types i.e., preemptive and non-preemptive systems. Though non-preemptive are simple when it comes to implementation, such policy loses its attraction when higher system utilization is desired. Due to the wider applicability and acceptance, preemptive systems have been investigated actively in the currently and a number of feasibility tests have been derived for single processor systems that can be easily extended to multi-processor systems [1-3, 11, 19].

The preemptive class of scheduling algorithms can be further classified into two major domains: (i) fixed priority, and (ii) dynamic priority [1, 2]. The main difference between both types is the priority assignment. Under fixed-priority algorithm, each task is assigned a unique priority that remains fixed as long the task set is under operation. In the dynamic priority techniques on the other hand, the priority of the task may change at run time and hence becomes unpredictable when the system becomes overloaded. In addition, fixed priority systems are simple from implementation perspective and can be easily implemented atop many available operating systems.

The problem of scheduling fixed systems was first addressed by Liu and Layland in 1973 [1] under simplified assumptions and authors derived the optimal static priority scheduling algorithm called rate monotonic (RM) algorithm for implicit-deadline model (when deadlines coincide with respective periods). Since then, to test the schedulablity of fixed priority scheduling system, there exist a number of feasibility tests [1, 4-10]. These tests can be partitioned into two broad categories i.e., Necessary and Sufficient Conditions (NSC), and Sufficient Conditions (SC). On one hand, NSC results in higher system utilization and can schedule real-time tasks in a system as long as the utilization is not more than 100%, while SC can promise system utilization up to the ln(2) only, where “n” is the number of tasks in the system. However, the complexity of NSC is pseudo-polynomial and restricts its use in online systems. On the other hand, SC are simple with O(n) complexity. Literature shows reveals that the lower complexity of SC comes at the price of lower system utilization while the complexity of NSC class is NP-hard in strong sense [11]. Consequently, variants of NSC have been proposed recently to lower the computational cost of feasibility tests instead of time complexity [3-10].

The NSC techniques can be divided into two major types (i) techniques based on scheduling points [3-7], and (ii) iterative solutions [7-8]. Under scheduling point’s techniques, authors in [4] restricted the feasibility of the system to be tested at a subset of scheduling points for reducing the computation cost of the system. Recently, [3] analyzed the same problem form the perspective of avoiding scheduling points which were unable to satisfy the requirement of higher priority tasks while checking feasibility of a task. Under iterative solutions [7-8], feasibility tests determine the response time of tasks in descending priority order. In such approaches, the duration from the released time of a task to finishing time is analyzed and as an outcome the task is declared schedulable, if response time is within the deadline, otherwise the task is decaled infeasible by RM scheduling policy. An improvement over [17] was made in [18] by obtaining a higher initial guess value that depends on the higher priority task. However, to the best of our knowledge, no work has been done for studying the impact of higher initial values for scheduling points based techniques. Extending the work done in [3-4], we propose a solution that allows the system to obtain higher values as the initial guess value while checking feasibility of a task. Our solution answers the schedulablity of the task set in a faster fashion without compromising the utilization of the system.

The remaining paper is organized as follows. Section 2 covers the background work and constructs the model to formulate our problem. The technique for obtaining higher initial value is described in Section 3, while Section 4 discusses experimental results. The paper is concluded in Section 5.

BACKGROUND AND PROBLEM FORMULATIONS

Before we discuss the background work, we introduce the task model that is used for deriving our main results in Section 3.

Let $\tau = \{\tau_1,...,\tau_n\}$ be a periodic task system, where each task τ_i is represented by its parameters execution time c_i, task period p_i, and deadline d_i. To formulate our model, we assume that the tasks are independent of each other and there is only a single processor available to schedule all the tasks in the system using RM. Being a fixed priority system, RM assigns static priorities on task activation rates (periods). For constrained deadline systems, when periods are larger than deadlines, an optimal priority technique was drawn in [13] called Deadline-Monotonic (DM) system. The RM and DM are identical when relative deadline of every task is proportional to its period. For simplicity, we assume implicit deadline model i.e., $p_i = d_i$. The utilization of task τ_i is defined as: $u_i = c_i / p_i$. The cumulative utilization u_{tot} of periodic task system τ is:

$$u_{tot} = \sum_{i=1}^{n} \frac{c_i}{p_i} \tag{1}$$

For validating timing constrains, feasibility tests– given a task set and system model, determining whether it is possible to meet all the deadlines– are performed to achieve system predictability [4-7, 14, 16]. The first feasibility test for RM was proposed in 1973 [6], by Liu and Leyland. According to [6], a periodic task system is schedulable if

$$u_{tot} \le n(2^{1/n} - 1) \tag{2}$$

Where n denotes the number of tasks in τ.

Equation 2 can only promise the feasibility of the system as long its utilizations less than 69% [1]. To overcome the theoretical difference in performance proposed by LL-bound, necessary and sufficient condition (NSC) based tests were proposed [1-9]. The feasibility can be either straight forward approaches [5-7] or iterative [8-9]. The straightforward solution test task feasibility only at times when tasks arrive, called scheduling points. The iterative techniques test task feasibility by employing iteration. Under both implementation, the time complexity remains pseudo-polynomial [11-12] and hence the focus is on reducing the computation cost of these techniques.

The workload due to τ_i at time t consists of its execution demand c_i as well as the interference it encounters due to higher priority tasks from τ_{i-1} to τ_1 and can be expressed mathematically as:

$$W_i(t) = c_i + \sum_{j=1}^{i-1} \left\lceil t/p_j \right\rceil c_j \tag{3}$$

A periodic task τ_i is feasible if we find some $t \in [0,t]$ satisfying

$$L_i = \min_{0<t\le p_i} (W_i(t) \le t) \tag{4}$$

In other words, task τ_i completes its computation requirements at time $t \in [0,t]$, if and only if the entire request from the $i-1$ higher priority tasks and computation time of τ_i is completed at or before time t. As t is a continuous variable, there are infinite numbers of points to be tested. The entire task set τ is feasible iff

$$L = \max_{1\le i\le n} \{ \min_{0<t\le p_i} \frac{W_i(t)}{t} \} \le 1 \tag{5}$$

The first attempt to limit the infinite number of points in interval $t \in [0,t]$ is made in [8]. The authors' show that $W_i(t)$ is constant, except at finite number of points, where tasks are released, called RM scheduling points. Consequently, to determine whether τ_i is schedulable, $W_i(t)$ is computed only at multiples of $\tau_i \le \tau_j, 1 \le j \le i$. Specifically, let

$$S_i = \left\{ ap_b \mid b = 1,...,i; a = 1,..., \lfloor p_i / p_b \rfloor \right\} \tag{6}$$

Under TDA, the fundamental theorem to determine whether an individual task is feasible or not.

Theorem 1. –Given a set of n periodic tasks $\tau_1,...,\tau_n$, τ_i can be feasibly scheduled for all tasks phasings using RM iff

$$L_i = \min_{t\in S_i} \frac{W_i(t)}{t} \le 1 \tag{7}$$

Theorem 1 is known as TDA [7]. To reduce the computation cost associated with TDA, authors in [4] proposed hyper-planes test. The Hyper-planes Exact Test (HET) reduces scheduling point for τ_i from set S_i to a reduced set $H_i(t)$. For any task τ_i, their test begins with p_i and expands its search space by

$$H_i(t) = H_{i-1}\left(\left\lfloor \frac{t}{p_i} \right\rfloor p_i \right) \cup H_{i-1}(t) \tag{8}$$

where H_0(t)={t}.

Furthermore, Theorem 1 was also extended in [15] by deriving a technique called Enhanced Time demand Analysis (ETDA).

Theorem2. Given a set of n periodic tasks { $\tau_1,...,\tau_n$ }, τ_i can be feasibly scheduled for all tasks phasings using RM *iff*

$$L_i = \min_{t \in Z_i} \frac{W_i(t)}{t} \leq 1 \qquad (9)$$

where $Z_i = S_i - X_{i-1}$ and X_{i-1} is the set of scheduling points at which the schedulability of T_{i-1} is negative. By extension $X_0 = \varnothing$.

It is evident from Inequality 9 that the TDA has been improved from computational perspective by restricting the scheduling points. Similarly, we extend the TDA by obtaining a higher initial value for any low priority task that help is avoiding many unnecessary steps.

HIGHER INITIAL VALUES FOR TASKS

In our approach, the system feasibility is tested in the highest priority first fashion where the test proceeds to the lower priority task only if the current periodic task is RM schedulable; otherwise the system is infeasible as per RM scheduling. With Theorem 1, the starting guess for a task τ_i is c_i and hence feasibility analysis starts with first scheduling point in the set of scheduling points S_i i.e., $p_1 \in S_i$: $\min_{t \in S_i}(W_i(t)/\mathrm{t} \leq 1)$. With this formulation, it is evident for a lower priority task τ_{i+1} that its workload $w_{i+1}(t)$ is again tested at p_1 while the demand at point p_1 is now higher than what was presented by τ_i at same point p_1. However, it cannot be concluded that the same p_1 that accommodates the workload of τ_i can also handle the workload due to τ_{i+1} as c_{i+1} is additional term contributing to the workload at p_1 by τ_{i+1}. This pattern suggests that τ_{i+1} has to be tested at p_1 and so on, unless the workload becomes less than or equal to the available time on a single processor system.

Let $w_{i,1}(t)$ is the first value for τ_i and $w_{i-1,j}(t)$ is the workload due to cumulative workload of lower priority tasks $\{\tau_1, \tau_2, ..., \tau_{i-1}\}$ which is feasibly at t. For $w_{i-1,j}(t)$, t is the first point $t \in S_i$ at which the schedulability of τ_{i-1} is answered. Since τ_i is unschedulable at t and hence $w_i(t) > \mathrm{t}$. Therefore, schedulability test now skips the remaining point in set as condition i.e., $\min\ t \in S_i$ is true.

We now explain the working of our solution in Table 1 which highlights the feasibility analysis of a task where the task computation and period may have random values such that computation demand of a task is not more than its respective deadline. Consider, Table 1 is being populated while analyzing the workload at multiple of higher tasks time periods, starting with highest priority first analysis approach, for a task set consisting of four tasks. As shown, the last value for any task τ_i becomes the first candidate value for the task τ_{i+1} at which its feasibility has to be tested. For task τ_2 in step#4, $w_2(t)$ is satisfied over point t_3 and hence $w_2(t)$ is declared $w_2^{'}(t)$. Consequently the starting value for τ_i becomes $w_3(t) = w_2^{'}(t)$.

Table 1: Feasibility analysis table of a four tasks set

Task#	Step#	S_i	Testing Point	Workload	$L_i \leq 1$
1	1	t_1	$t_1 \in S_1$	$w_1(\mathrm{t}_1)$	✓
2	2	t_1, t_2, t_3	$t_1 \in S_2$	$w_2(\mathrm{t}_1)$	×
	3	t_1, t_2, t_3	$t_2 \in S_2$	$w_2(\mathrm{t}_2)$	×
	4	t_1, t_2, t_3	$t_3 \in S_2$	$w_2(\mathrm{t}_3)$	✓
3	5	t_1, t_2, t_3, t_4	$t_1 \in S_3$	$w_3(\mathrm{t}_1)$	×
	6	t_1, t_2, t_3, t_4	$t_2 \in S_3$	$w_3(\mathrm{t}_2)$	×
	7	t_1, t_2, t_3, t_4	$t_3 \in S_3$	$w_3(\mathrm{t}_3)$	×
	8	t_1, t_2, t_3, t_4	$t_4 \in S_3$	$w_3(\mathrm{t}_4)$	✓
4	9	t_1, t_2, t_3, t_4, t_5	$t_1 \in S_4$	$w_4(\mathrm{t}_1)$	×
	10	t_1, t_2, t_3, t_4, t_5	$t_2 \in S_4$	$w_4(\mathrm{t}_2)$	×
	11	t_1, t_2, t_3, t_4, t_5	$t_3 \in S_4$	$w_4(\mathrm{t}_3)$	×
	12	t_1, t_2, t_3, t_4, t_5	$t_4 \in S_4$	$w_4(\mathrm{t}_4)$	×
	13	t_1, t_2, t_3, t_4, t_5	$t_5 \in S_4$	$w_4(\mathrm{t}_5)$	✓

With every point t in set S_i, the workload is non-decreasing function of cumulative workload as higher priority tasks instances may arrive in interval $[0, t]$. This observation suggests that any task τ_i that is non-schedulable at point t reveals that τ_{i+1} is also not schedulable at t due to additional computation associated with τ_{i+1}. The workload that is satisfied at t for τ_i is $w_i(t)$, so for τ_{i+1}, it becomes the initial value. With same argument the workload at any point t is:

$$w_i(t) < w_{i+1}(t) < ... < w_n(t)$$

and hence:

$$w_i^{'}(t) < w_{i+1}^{'}(t) < ... < w_n^{'}(t) \qquad (10)$$

We now apply Inequality 10 to the task set provided in Table 1. Our formulation provides the initial values for task τ_2 in step#4 and hence feasibility test skips step#5-7. For the fourth task τ_4, it skips 4 points moreand so on. The advantage of our test becomes more visible when applied to larger tasks sets as higher initial values are obtained for the lower priority tasks and hence the test feasibility is determined much faster. We represent this scheme by Higher Time Demand Analysis (HTDA).

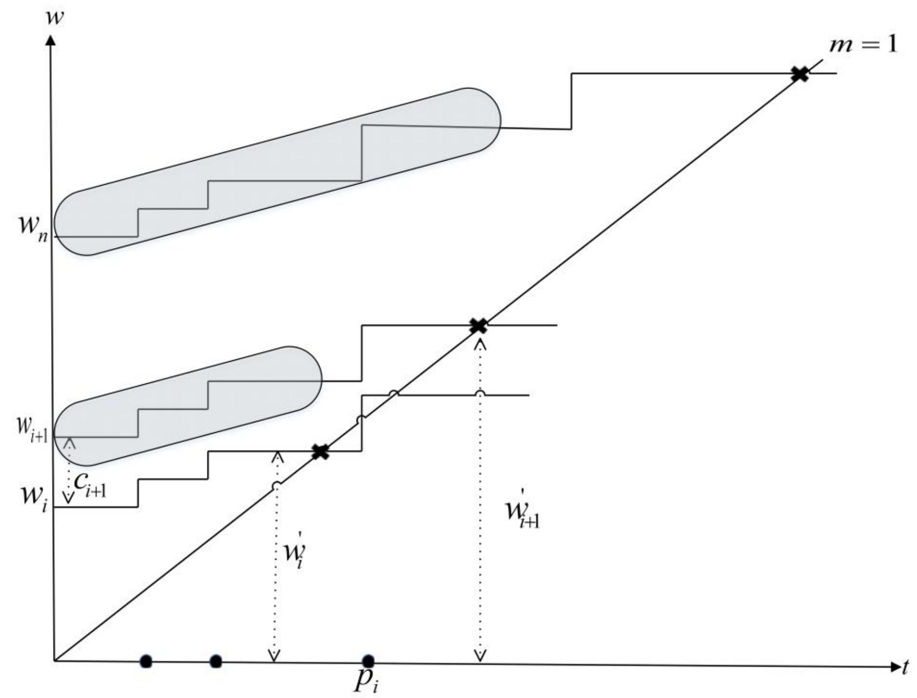


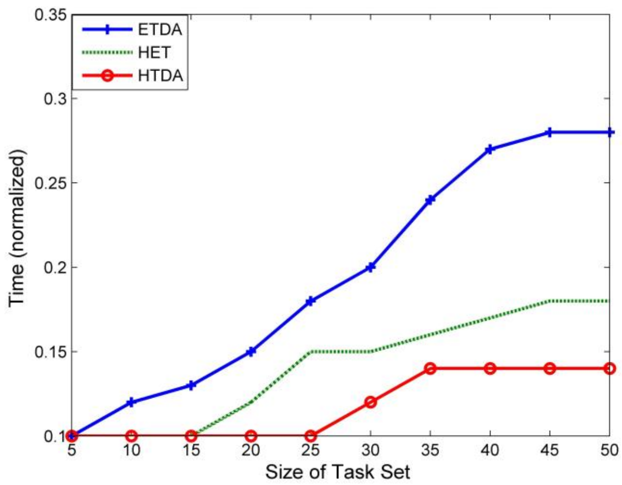
Figure 1: Initial starting value for a low priority task

Figure 1 provides the graphical presentation of HTDA. We plot the time demand of tasks and identify the scheduling point that accommodates the workload presented by a task Ti. The x-axis represents the time while y-axis shows the demand of tasks at a given point in time. The region below the line having slope 1 is feasible for any task according to RM our approach. The dotted lines represent the workload of individual tasks that are non-decreasing and monotonically increase at task periods of all higher priority tasks. It can be seen in Figure 1 that the jump between $w_i(t)$ and $w_{i+1}(t)$ at any point in time t is least c_{i+1} units. The think dots on x-axis represent the time where the workload changes and cross highlights the first feasible scheduling point for a task where workload is satisfied at the given point. The feasibility of task τ_i becomes true at point reflected with cross. The shaded region identifies the values that are unnecessary for lower priority task τ_{i+1} and hence should ignore as they are obviously not going to address the demand. This shaded region constitutes the initial value for the next lower priority task and larger is this region, the better it would be for the lower priority task. For testing the feasibility of lower priority task τ_{i+1}, a higher value is assigned to τ_{i+1} as initial cumulative demand. This is presented by $w_i^{'}$ and $w_{i+1}^{'}$ for task τ_i and τ_{i+1}, respectively.

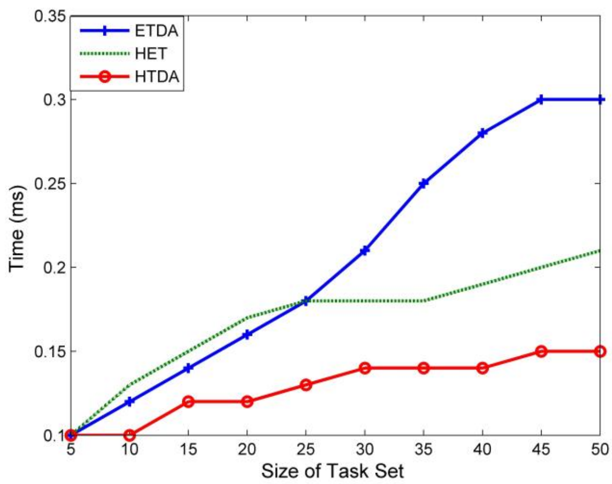
EXPERIMENTAL RESULTS AND ANALYSIS

To align with previous techniques presented in literature, we evaluate the performance of HTDA and compare the results with ETDA and HET from run time perspective.. We generate random task periods from 10-100 tasks with step size of 5 tasks. In our task set generation module, no tasks have the same task period and periods are in the range of $[10,1000]$ with uniform distribution. Random values are taken for corresponding task execution demands within $[1, p_i]$. For task computation values, we use uniform distribution. The priority of individual task has been assigned as per RM assignment criteria. Experimentation is done in MATLAB and running on a PC with 3.40GHz Intel (i7-3770) and 8GB RAM under Linux. We only analyze the run time performance as a rule of thumb, as the time taken to solve feasibility problem is the simple criteria for evaluating the performance of a given algorithm

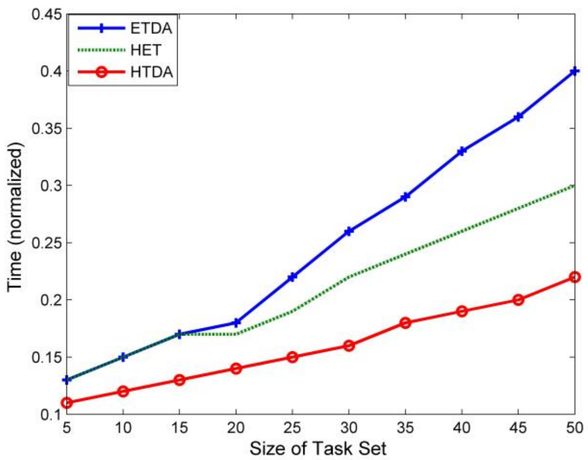
The performance of all techniques is better for the task sets when the number of task is low and increases as the size increases. When system utilization is 80% then its very likely that the cumulative demand is fulfilled with testing a few scheduling points and that shown in Figure 2(a). Even under 80% utilization, the increased number of task present more load for the test. Since ETDA needs to maintain a list of previously tested scheduling points and hence slower as compared to HET or HDTA. Similarly, the HET perform union operation while testing feasibility using recursive approach and hence the commutation cost is more than HTDA but lower than ETDA. Due to this recursive nature, HET is behaving similar to the ETDA when system utilization is 85% or 90%, as this is the utilization at which the feasibility has to be test maximum tasks and its very likely that the lowest priority task is also schedulable per RM algorithm. Irrespective of utilization, HET has to confine the search space to a set number of scheduling points while HTDA just proceed with higher initial value. This trend is shown in Figure 2(a) to (d). Even when system utilization is 95%, HTDA outclass existing techniques due to its straight forward approach and under such utilization; it is very likely some tasks can miss the deadline. For lower utilization and less number of task, HTDA is very efficient as only points are analyzed while testing feasibility of a task. The worst case scenario is Figure 2(d) when system utilization is 95% and even for less number of task, nor scheduling point have to be analyzed which is aligned to our formulation.



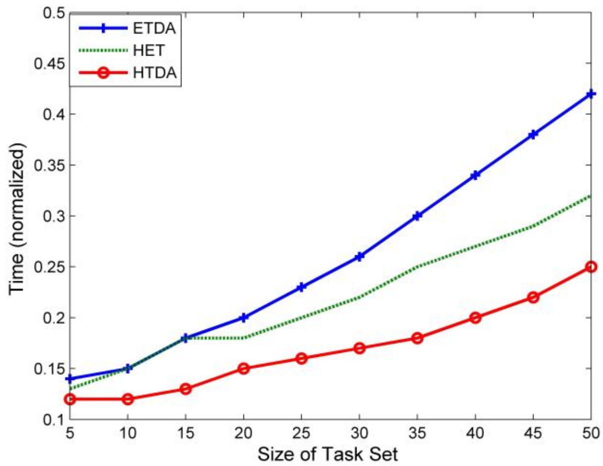
(a) System utilization = 80%



(b) System utilization = 85%



(c) System utilization = 90%



(d) System utilization = 95%

Figure 2: Performance analysis under varying system utilization

CONCLUSION

The problem of analyzing the feasibility of periodic task under RM scheduling algorithm is discussed and a pattern between the two consecutive tasks schedulability is identified. The relationship between two neighboring tasks showed that any scheduling point which satisfies the CPU demand of a task becomes the initial value for the lower priority task at which the feasibility can be tested. The aforementioned relationship was exploited for faster feasibility analysis of time demand analysis for RM schedulability. Experimental results confirmed that the proposed method is quite competitive as compared to existing counterparts.

REFERENCES

[1] 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

[2] J. W. S. Liu, Real Time Systems, Prentice Hall, 2000.

[3] N. Min-Allah, S. U. Khan and Wang Yongji, “ Optimal Task Execution Times for Periodic Tasks Using Nonlinear Constrained Optimization", Journal of Supercomputing, pp. 1--19, 2010.

[4] E. Bini, G. C. Buttazzo, "The Space of Rate Monotonic Schedulability", Proceedings of the 23th IEEE Real-Time Systems Symposium, Austin, Texas, pp.169-177, 2002.

[5] E. Bini, G. C. Buttazzo, "Schedulability Analysis of Periodic Fixed Priority Systems", IEEE Transactions on Computers, Vol.53, No.11, pp.1462-1473, 2004.

[6] J. Y. T. Leung and J. Whitehead, "On the Complexity of Fixed-Priority Scheduling of Periodic", Real-Time Tasks Performance Evaluation. NO.2, pp.237-250, 1982.

[7] J. P. Lehoczky, L. Sha, Y. Ding, "The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior", Proceedings of the IEEE Real-Time System Symposium, pp.166-171, 1989.

[8] N. C. Audsley, A. Burns, K. Tindell, A. Wellings," Applyingnew scheduling theory to static priority preemptive scheduling", Software Engineering Journal, 1993.

[9] M. Sj¨odin and H. Hansson, "Improved response-time analysis calculations", Proceedings of the 19th IEEE Real-Time Systems Symposium, pp.399-409, 1998.

[10] 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(1), pp. 48-57, 2012.

[11] Z. Gu, M. Yuan and X. He, "Optimal Static Task Scheduling on Reconfigurable Hardware Devices Using Model-Checking", In proceedings of the 13th IEEE Real Time and Embedded Technology and Applications Symposium, pp.32-44, 2007.

[12] K. W. Tindell, A. Bums, A. J. Wellings, "An extendible approach for analyzing fixed priority hard real-time tasks", Real-Time Systems Journal, No.6, pp.133-151, 1994.

[13] J. Y. T. Leung and J. Whitehead, "On the Complexity of Fixed-Priority Scheduling of Periodic", Real-Time Tasks Performance Evaluation. NO.2, pp.237-250, 1982

[14] M. Joseph and P. Pandya, "Finding response times in a real-time system", The Computer Journal, Vol.29, No.5, pp.390-395, 1986.

[15] N. Min-Allah,Yong-Ji Wang, Jian-Sheng Xing, Enhanced Rate Monotonic Time Demand Analysis , IJCSES International Journal of Computer Sciences and Engineering Systems, 2007.

[16] J. P. Lehoczky, L. Sha, Y. Ding, "The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior", Proceedings of the IEEE Real-Time System Symposium, pp.166-171, 1989.

[17] N. C. Audsley, A. Burns, K. Tindell, A. Wellings, " Applyingnew scheduling theory to static priority preemptive scheduling", Software Engineering Journal, 1993.

[18] M. Sj¨odin and H. Hansson, "Improved response-time analysis calculations", Proceedings of the 19th IEEE Real-Time Systems Symposium, pp.399-409, 1998.

[19] S. Alrashed, Reducing power consumption of non-preemptive real-time Systems, The Journal of Supercomputing 73 (12), 5402-5413, 2017.

[20] M.B. Qureshi, S. Alrashed, N. Min-Allah, J. Kołodziej, P. Arabas, Maintaining the Feasibility of Hard Real–Time Systems with a Reduced Number of Priority Levels, International Journal of Applied Mathematics and Computer Science, 25(4), pp. 709-722, 2015.